IN-34
191156
p-282

NASA Technical Memorandum 106341

Proteus Three-Dimensional Navier-Stokes Computer Code–Version 1.0

Volume 3–Programmer's Reference

Charles E. Towne, John R. Schwab, and Trong T. Bui
Lewis Research Center
Cleveland, Ohio

October 1993

NASA

CONTENTS

PROTEUS THREE-DIMENSIONAL NAVIER-STOKES COMPUTER CODE - VERSION 1.0

Volume 3 - Programmer's Reference

Charles E. Towne, John R. Schwab, Trong T. Bui

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

SUMMARY

A computer code called *Proteus* has been developed to solve the three-dimensional, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized.

The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit solution procedure with generalized first- or second-order time differencing. The boundary conditions are also treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model.

The program contains many operating options. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple Cartesian or cylindrical grids may be generated internally by the program. More complex geometries require an externally generated computational coordinate system.

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in *Proteus*. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2 is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3, the current volume, is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

A two-dimensional/axisymmetric version of *Proteus* code also exists, and was originally released in late 1989.

1.0 INTRODUCTION

Much of the effort in applied computational fluid dynamics consists of modifying an existing program for whatever geometries and flow regimes are of current interest to the researcher. Unfortunately, nearly all of the available non-proprietary programs were started as research projects with the emphasis on demonstrating the numerical algorithm rather than ease of use or ease of modification. The developers usually intend to clean up and formally document the program, but the immediate need to extend it to new geometries and flow regimes takes precedence.

The result is often a haphazard collection of poorly written code without any consistent structure. An extensively modified program may not even perform as expected under certain combinations of operating options. Each new user must invest considerable time and effort in attempting to understand the underlying structure of the program if intending to do anything more than run standard test cases with it. The user's subsequent modifications further obscure the program structure and therefore make it even more difficult for others to understand.

The *Proteus* three-dimensional Navier-Stokes computer program is a user-oriented and easily-modifiable flow analysis program for aerospace propulsion applications. Readability, modularity, and documentation were primary objectives during its development. The entire program was specified, designed, and implemented in a controlled, systematic manner. Strict programming standards were enforced by immediate peer review of code modules; Kernighan and Plauger (1978) provided many useful ideas about consistent programming style. Every subroutine contains an extensive comment section describing the purpose, input variables, output variables, and calling sequence of the subroutine. With just three clearly-defined exceptions, the entire program is written in ANSI standard Fortran 77 to enhance portability. A master version of the program is maintained and periodically updated with corrections, as well as extensions of general interest (e.g., turbulence models.)

The *Proteus* program solves the unsteady, compressible, Reynolds-averaged Navier-Stokes equations in strong conservation law form. The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit (ADI) scheme with generalized time and space differencing (Briley and McDonald, 1977; Beam and Warming, 1978). Turbulence is modeled using either the Baldwin and Lomax (1978) algebraic eddy-viscosity model or the Chien (1982) two-equation model. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. The boundary conditions are treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available.

The program contains many operating options. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple grids may be generated internally by the program; more complex geometries require external grid generation, such as that developed by Chen and Schwab (1988).

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in *Proteus*. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2 is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and se-

veral test cases. Volume 3, the current volume, is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

A two-dimensional/axisymmetric version of *Proteus* code also exists, and was originally released in late 1989 (Towne, Schwab, Benson, and Suresh, 1990).

The authors would like to acknowledge the significant contributions made by their co-workers. Tom Benson provided part of the original impetus for the development of *Proteus*, and did the original coding of the block tri-diagonal inversion routines. Simon Chen did the original coding of the Baldwin-Lomax turbulence model, and consulted in the implementation of the nonlinear coefficient artificial viscosity model. William Kunik developed the original code for computing the metrics of the generalized nonorthogonal grid transformation. Frank Molls has created a separate diagonalized version of the code. Ambady Suresh did the original coding for the second-order time differencing and for the nonlinear coefficient artificial viscosity model. These people, along with Dick Cavicchi, Julie Conley, Jason Solbeck, and Pat Zeman, have also run many debugging and verification cases.

2.0 PROGRAM STRUCTURE

2.1 FLOW CHART

In this section, a flow chart is presented showing the overall sequence of tasks performed by the three-dimensional *Proteus* computer code. Depending on the various options used in a particular run, of course, some of the elements in the chart may be skipped.

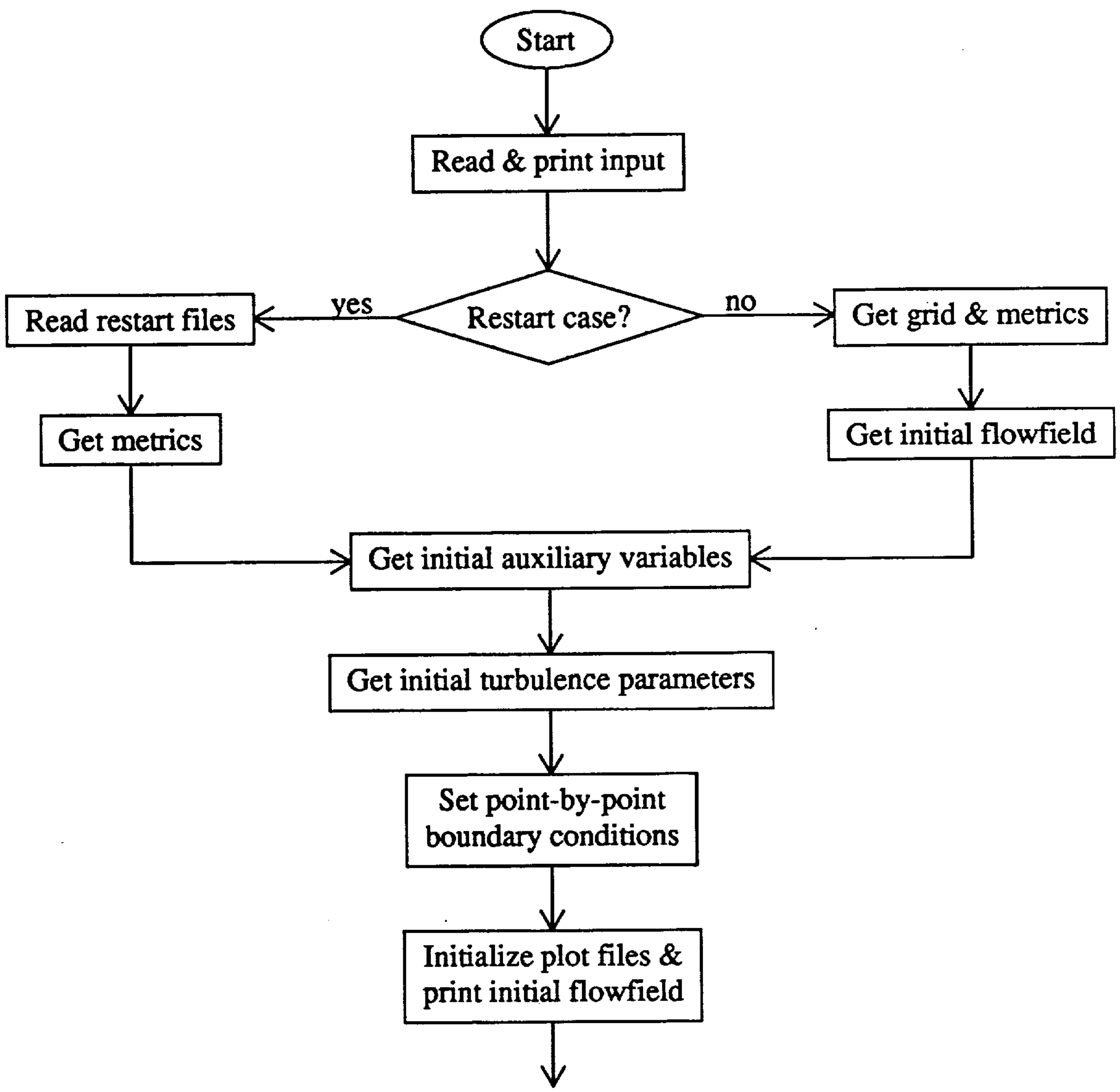

Figure 2.1 - Flow chart for the 3-D Proteus computer code.

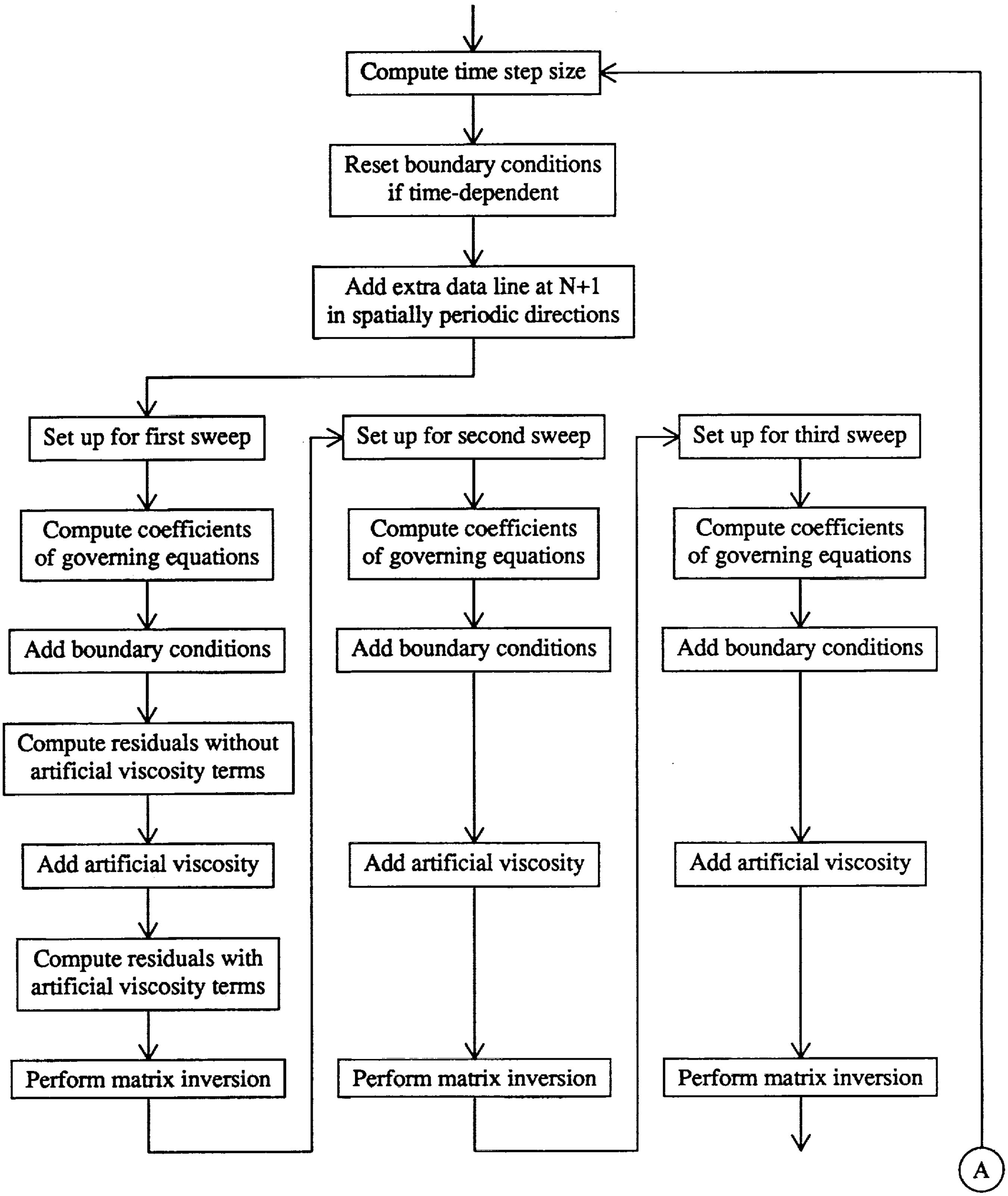

Figure 2.1 - Continued.

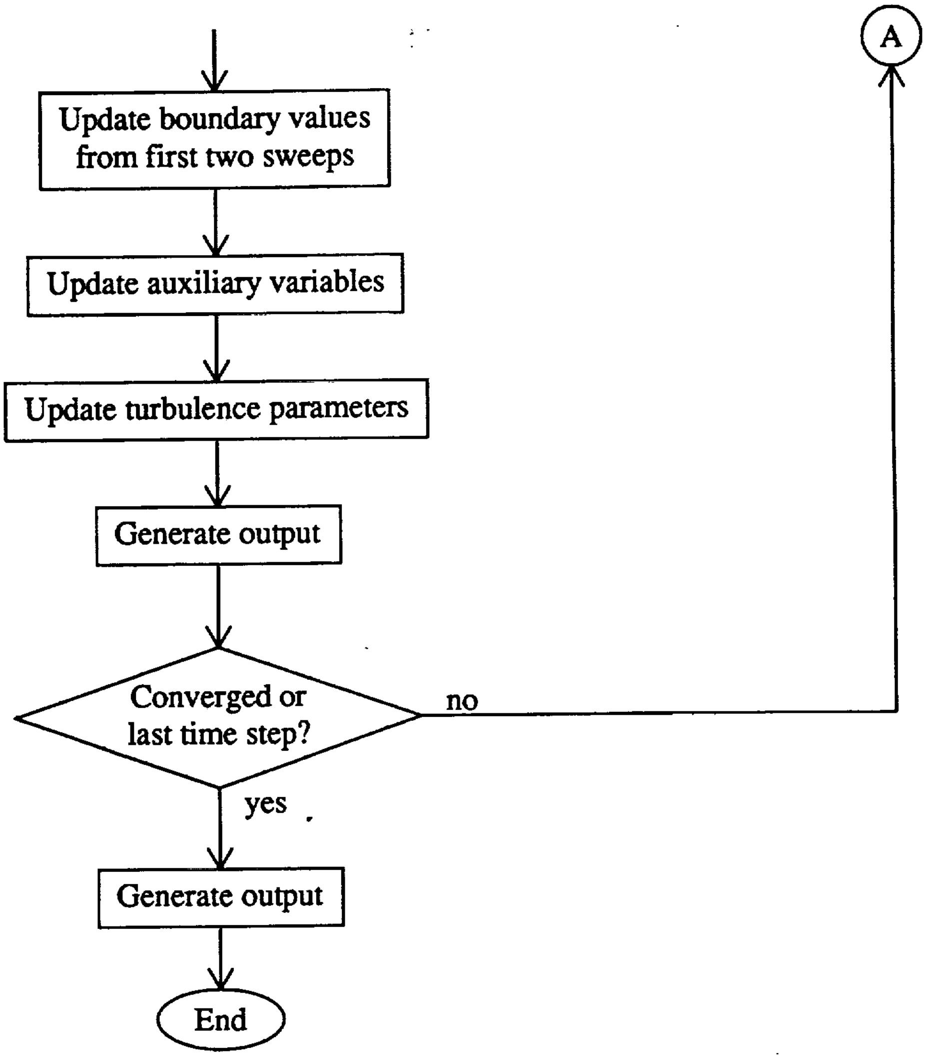

Figure 2.1 - Concluded.

2.2 SUBPROGRAM CALLING TREE

In this section, the calling sequence for the various subprograms in the *Proteus* 3-D code is shown using a tree structure. The subheadings correspond to the elements of the flow chart shown in the previous section. The main program, listed in the first column, calls the subprograms in the second column, which in turn call those in the third column, etc.[1] Since some Cray library routines are called multiple times, only the first call from a subprogram is shown for these routines. For any given case, of course, some of the subprograms shown will not be used. The subprograms needed for a particular case will depend on the combination of input parameters being used. The individual subprograms are described in detail in Section 4.0.

INITIALIZATION					
Read and print input.					
MAIN	INPUT	ISAMAX			
Get grid and metric parameters.					
MAIN	GEOM	PAK	ROBTS		
			CUBIC		
		METS	OUTPUT	PRTOUT	
Get initial flow field.					
MAIN	INITC	REST	METS		
		INIT			
		FTEMP			
		EQSTAT			
		TURBBL	BLOUT	VORTEX	
				ISRCHFLT	
				ISRCHFGT	
				ISAMIN	
				ISAMAX	
				WHENFLT	
				GATHER	
			BLIN	ISRCHFGT	
				VORTEX	
		KEINIT	TURBBL	BLOUT	VORTEX
					ISRCHFLT
					ISRCHFGT
					ISAMIN
					ISAMAX
					WHENFLT
					GATHER
				BLIN	ISRCHFGT
					VORTEX
			YPLUSN	VORTEX	
			PRODCT		
		YPLUSN	VORTEX		
Set point-by-point boundary condition values.					
MAIN	BCSET				

[1] Throughout this Programmer's Reference, elements of the Fortran language, such as input variables and subprogram names, are printed in the text using uppercase letters. However, in most implementations, Fortran is case-insensitive. The *Proteus* source code itself is written in lowercase.

Initialize plot files and print initial or restart flow field.					
MAIN	PLOT OUTPUT OUTW	 VORTEX PRTOUT			
SET UP FOR TIME STEP					
Compute time step size.					
MAIN	TIMSTP	ISAMAX			
Reset boundary conditions if time-dependent.					
MAIN	TBC				
FILL BLOCK COEFFICIENT MATRIX					
Add extra data line at $N + 1$ if spatially periodic in sweep direction.					
MAIN	EXEC	PERIOD			
Compute coefficients of governing equations.					
MAIN	EXEC	EQSTAT COEFC COEFX COEFY COEFZ COEFE1 COEFE2			

Add boundary conditions.					
MAIN	EXEC	EQSTAT			
		BCGEN	BCQ	BCMET	
				BCGRAD	
			BCUVEL	BCMET	
				BCGRAD	
			BCVVEL	BCMET	
				BCGRAD	
			BCWVEL	BCMET	
				BCGRAD	
			BCPRES	BCMET	
				BCGRAD	
			BCTEMP	BCMET	
				BCGRAD	
			BCDENS	BCMET	
				BCGRAD	
			BCNVEL	BCVN	
				BCMET	
			BC1VEL	BCIMET	
				BCV1	
				BCMET	
			BC2VEL	BCIMET	
				BCV2	
				BCMET	
			BC3VEL	BCIMET	
				BCV3	
				BCMET	
			BCF	BCFLIN	
				BCMET	
			ISRCHEQ		
			BLKOUT		
		BCELIM	SGEFA		
			SGESL		
Compute residuals without artificial viscosity terms (sweep 1 only.)					
MAIN	EXEC	RESID	SNRM2		
			ISAMAX		
			SASUM		
Add artificial viscosity.					
MAIN	EXEC	AVISC1	BLKOUT		
		AVISC2	BLKOUT		
Compute residuals with artificial viscosity terms (sweep 1 only.)					
MAIN	EXEC	RESID	SNRM2		
			ISAMAX		
			SASUM		

SOLVE DIFFERENCE EQUATIONS					
Perform matrix inversion.					
MAIN	EXEC	ADI	BLKOUT		
			BLK4P		
			BLK4	FILTER	ISAMAX
					ISRCHEQ
					BLKOUT
			BLK5P		
			BLK5	FILTER	ISAMAX
					ISRCHEQ
					BLKOUT
		UPDATE			
Update boundary values from first two sweeps.					
MAIN	EXEC	BVUP	EQSTAT		
			BCGEN	BCQ	BCMET
					BCGRAD
				BCUVEL	BCMET
					BCGRAD
				BCVVEL	BCMET
					BCGRAD
				BCWVEL	BCMET
					BCGRAD
				BCPRES	BCMET
					BCGRAD
				BCTEMP	BCMET
					BCGRAD
				BCDENS	BCMET
					BCGRAD
				BCNVEL	BCVN
					BCMET
				BC1VEL	BCIMET
					BCV1
					BCMET
				BC2VEL	BCIMET
					BCV2
					BCMET
				BC3VEL	BCIMET
					BCV3
					BCMET
				BCF	BCFLIN
					BCMET
				ISRCHEQ	
				BLKOUT	
			SGEFA		
			SGESL		
FINISH TIME STEP AND CHECK RESULTS					
Update auxiliary variables.					
MAIN	EQSTAT				
	FTEMP				

Update turbulence parameters.					
MAIN	TURBBL	BLOUT	VORTEX		
			ISRCHFLT		
			ISRCHFGT		
			ISAMIN		
			ISAMAX		
			WHENFLT		
			GATHER		
		BLIN	ISRCHFGT		
			VORTEX		
	TURBCH	YPLUSN	VORTEX		
		PRODCT			
		EXECT	PERIOD		
			SWUP		
			SWDOWN		
			UPDTKE		
Check for convergence, and get CPU time remaining.					
MAIN	CONV	ISAMAX			
	TREMAIN				
GENERATE OUTPUT					
Print flow field output.					
MAIN	OUTPUT	VORTEX			
		PRTOUT			
	OUTW				
Write plot and restart files.					
MAIN	PLOT				
	REST				
Print convergence history.					
MAIN	PRTHST				

2.3 PROGRAMMING CONVENTIONS AND NOTES

2.3.1 Computer & Language

At NASA Lewis Research Center, *Proteus* is normally run on a Cray X-MP or Y-MP computer. With just three known exceptions, it is written entirely in ANSI standard Fortran 77 as described in the *CF77 Compiling System, Volume 1: Fortran Reference Manual* (Cray Research, Inc., 1990). The first exception is the use of namelist input. With namelist input, it's relatively easy to create and/or modify input files, to read the resulting files, and to program default values. Since most Fortran compilers allow namelist input, its use is not considered a serious problem. The second exception is the use of *CALL statements to include *COMDECKS, which contain the labeled common blocks, in most of the subprograms. This is a Cray UPDATE feature, and therefore the source code must be processed by UPDATE to create a file that can be compiled.[2] UPDATE is described in the *UPDATE Reference Manual* (Cray Research, Inc., 1988). Since using the *CALL statements results in cleaner, more readable code, and since many computer systems have an analogous feature, the *CALL statements were left in the program. The third exception is the use of lowercase alphabetic characters in the Fortran source code. This makes the code easier to read, and is a common extension to Fortran 77.

---

[2] See the example in Section 8.1 of Volume 2.

Several library subroutines are called by *Proteus*. SGEFA and SGESL are Cray versions of LINPACK routines. SASUM and SNRM2 are Cray Basic Linear Algebra Subprograms (BLAS). GATHER is a Cray Linear Algebra routine. ISAMAX, ISAMIN, ISRCHEQ, ISRCHFGT, ISRCHFLT, and WHENFLT are Cray search routines. TREMAIN is a Cray Fortran library routine. All of these routines are described in detail in Section 4.0 of Volume 3. In addition, SGEFA and SGESL are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b) and by Dongarra, Moler, Bunch, and Stewart (1979); SASUM, SNRM2, GATHER, ISAMAX, ISAMIN, ISRCHEQ, ISRCHFGT, ISRCHFLT, and WHENFLT are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b); and TREMAIN is described in *Volume 1: UNICOS Fortran Library Reference Manual* (Cray Research, Inc., 1989a).

The *Proteus* code is highly vectorized for optimal performance on the Cray. The coefficient generation is vectorized in the ADI sweep direction. Since the coefficient matrix is block tridiagonal, the equations are solved using the Thomas algorithm. This algorithm is recursive, and therefore cannot be vectorized in the sweep direction. However, by storing the coefficients and source terms in all three coordinate directions, the algorithm can be vectorized in one of the non-sweep directions. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

2.3.2 Fortran Variables

Variable Names

In developing *Proteus*, code readability has been emphasized. We have therefore attempted to choose Fortran variable names that are meaningful. In general, they either match the notation used in the analysis description in Volume 1, or are in some way descriptive of the parameter being represented. For example, RHO, U, V, W, and ET are the Fortran variables representing the density ρ, the velocities u, v, and w, and the total energy per unit volume E_T.

Real and Integer Variables

In general, the type (real or integer) of the Fortran variables follows standard Fortran convention (i.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.) There are, however, several variables that would normally be integer but are explicitly declared to be real. These are noted in the input description in Section 3.0 of Volume 2, and in the description of common block variables in Section 3.0 of this volume.

Array Dimensions

Most Fortran arrays are dimensioned using dimensioning parameters. These parameters are set in COMDECK PARAMS1. This allows the code to be re-dimensioned simply be changing the appropriate parameters, and then recompiling the entire program. The dimensioning parameters are described in Section 6.2 of Volume 2.

Initialization

All of the input Fortran variables, plus some additional variables, are initialized in BLOCK DATA. Most of the input variables are initialized to their default values directly, but some are initialized to values that trigger the setting of default values in subroutine INPUT. On the Cray X-MP and Y-MP at NASA Lewis, all uninitialized variables have the value zero. There are no known instances in the *Proteus* code, however, in which a variable is used before it is assigned a value.

Nondimensionalization

In general, Fortran variables representing physical quantities, such as RHO, U, etc., are nondimensional. Two types of nondimensionalizing factors are used - *reference* conditions and *normalizing* conditions. The factors used to nondimensionalize the governing equations in Section 2.0 of Volume 1 are called *normalizing* conditions. These normalizing conditions are defined by six basic *reference* conditions, for length, velocity, temperature, density, viscosity, and thermal conductivity, which are specified by the user. The normalizing conditions used in *Proteus* are listed in Table 3-1 of Volume 2.

Note that for some variables, like pressure, the normalizing condition is dictated by the form of the governing equations once the six basic reference conditions are chosen. Unfortunately, some of these may not be physically meaningful or convenient for use in setting up input conditions. Therefore, some additional reference conditions are defined from the six user-supplied ones. The reference conditions are listed in Table 3-2 of Volume 2.

Throughout most of the *Proteus* code, physical variables are nondimensionalized by the normalizing conditions. For input and output, however, variables are nondimensionalized by the reference conditions because they are usually more physically meaningful for the user. The Fortran variables representing the reference conditions themselves are, of course, dimensional.

One-Dimensional Addressing of Three-Dimensional Arrays

In the solution algorithm used in *Proteus*, there are several instances in which the same steps must be followed in all three ADI sweep directions. An example is the computation, in the COEFC, COEFX, COEFY, COEFZ, and COEFE1 routines, of the submatrices in the block tridiagonal coefficient matrix. These computations involve various flow variables, such as RHO, U, etc., and metric quantities, such as XIX, XIY, etc. These are stored as three-dimensional arrays, with the three subscripts representing, in order, the indices in the computational ξ, η, and ζ directions. For the first ADI sweep, values at various ξ indices are needed at fixed η and ζ indices. For the second ADI sweep, values at various η indices are needed at fixed ξ and ζ indices. And for the third ADI sweep, values at various ζ indices are needed at fixed ξ and η indices. In order to use the same coding for all three sweeps, a scheme for one-dimensional addressing of a three-dimensional array has been used.[3]

In Fortran, multi-dimensional arrays are actually stored in memory as a one-dimensional sequence of values, with the first subscript incremented over its range first, then the second subscript, etc. We take advantage of this in *Proteus*. As a first step, the three-dimensional array is equivalenced to a one-dimensional array of the same total length. The one-dimensional array name is derived from the three-dimensional array name by adding a "1". Thus, letting F represent a typical three-dimensional array,

```
dimension f(n1p,n2p,n3p),f1(ntotp)
equivalence (f(1,1,1),f1(1))
```

where N1P, N2P, and N3P are dimensioning parameters specifying the dimension size in the ξ, η, and ζ directions, and NTOTP is a dimensioning parameter equal to $N1P \times N2P \times N3P$. Next, we define a "step factor", which depends on the ADI sweep, and a "base index" which depends on the indices in the non-sweep directions. For the first ADI sweep,

```
      istep = 1
      do 1000 i3 = 2,npt3-1
      do 1000 i2 = 2,npt2-1
      iv = i2
      ibase = 1 + (i2-1)*n1p + (i3-1)*n1p*n2p
           .
           .
           .
 1000 continue
```

For the second ADI sweep,

```
      istep = n1p
      do 2000 i3 = 2,npt3-1
```

[3] An alternative would be to switch the order of the three subscripts in these arrays after each sweep. Since these arrays are used in many other areas of the code, this idea was discarded as being unnecessarily confusing. It should be noted, however, that there are some arrays in *Proteus* in which the order of the first two subscripts does switch between ADI sweeps. These are the A, B, C, and S arrays, which represent the coefficient submatrices and the source term subvector, and the METX, METY, METZ, and METT arrays, which represent the metric coefficients in the sweep direction. For these arrays, the first subscript is the index in one of the non-sweep directions (i.e., the η direction for the first sweep and the ξ direction for the second and third sweeps), and the second is the index in the sweep direction (i.e., ξ for the first sweep, η for the second sweep, and ζ for the third sweep.)

```
      do 2000 i1 = 2,npt1-1
      iv = i1
      ibase = i1 + (i3-1)*n1p*n2p
                    .
                    .
                    .
 2000 continue
```

And for the third ADI sweep,

```
      istep = n1p*n2p
      do 3000 i2 = 2,npt2-1
      do 3000 i1 = 2,npt1-1
      iv = i1
      ibase = i1 + (i2-1)*n1p
                    .
                    .
                    .
 3000 continue
```

In all of the above examples, the inner loop is in one of the non-sweep directions and IV therefore represents an index in one of the non-sweep directions. Nested inside this loop is a third loop, in the sweep direction. In this innermost loop, we can compute the equivalent one-dimensional address for a location in a three-dimensional array from the step factor, the base index, and the index in the sweep direction. Thus, for any of the ADI sweeps, the innermost loop looks like

```
      do 100 i = 2,npts-1
      iim1 = ibase + istep*(i-2)
      ii   = ibase + istep*(i-1)
      iip1 = ibase + istep*i
                    .
                    .
                    .
  100 continue
```

where I represents the index in the sweep direction. With this coding, for the first sweep

```
      f1(iim1) = f(i1-1,i2,i3)
      f1(ii  ) = f(i1  ,i2,i3)
      f1(iip1) = f(i1+1,i2,i3)
```

For the second sweep,

```
      f1(iim1) = f(i1,i2-1,i3)
      f1(ii  ) = f(i1,i2  ,i3)
      f1(iip1) = f(i1,i2+1,i3)
```

And for the third sweep,

```
      f1(iim1) = f(i1,i2,i3-1)
      f1(ii  ) = f(i1,i2,i3  )
      f1(iip1) = f(i1,i2,i3+1)
```

Two-Level Storage

With the Beam-Warming time differencing scheme used in *Proteus*, the dependent variables RHO, U, V, W, and ET must be stored at two time levels. For convenience, T is also stored at two time levels. In the ADI solution procedure, RHO, U, etc. are at the known time level n. The corresponding variable at the other time level is denoted by adding an "L" to the variable name. Exactly which time level the "L" variable is at depends on the stage in the solution procedure. Letting F represent one of these variables, the time levels for F and FL are listed in the following table for the different stages of the solution procedure. Recall that * and ** represent the intermediate time levels after the first and second ADI sweeps.

STAGE IN TIME STEP FROM LEVEL n TO $n+1$	TIME LEVEL FOR F	TIME LEVEL FOR FL
From start to end of sweep 1	n	$n-1$
From end of sweep 1 to end of sweep 2	n	*
From end of sweep 2 to end of sweep 3	n	**
From end of sweep 3 to update in EXEC	n	$n+1$
From update in EXEC to start of next step	$n+1$	n

DUMMY Array

For convenience, a three-dimensional array called DUMMY is stored in common block DUMMY1 and used as a temporary storage location in several areas of the code. This array is dimensioned N1P by N2P by N3P, the same as the flow variables, metrics, etc. DUMMY is used internally in subroutines BLIN, BLOUT, CONV, and RESID. It is also defined in subroutine YPLUSN for use in subroutines SWDOWN, SWUP, and KEINIT. And finally, it is defined in subroutine OUTPUT and passed as an argument into subroutine PRTOUT. Details on its use are presented in the subroutine descriptions in Section 4.0.

3.0 COMMON BLOCKS

Transfer of data between routines in *Proteus* is primarily accomplished through the use of labeled common blocks. Each common block contains variables dealing with a particular aspect of the analysis, and is stored in a separate Cray COMDECK (Cray Research, Inc., 1988). The common block names are the same as the COMDECK names. These names also correspond to the names of the input namelists. All the variables in namelist BC are stored in common block BC1, etc. The Fortran variables in each common block are stored in alphabetical order.

3.1 COMMON BLOCK SUMMARY

Block Name	Description
BC1	Boundary condition parameters for the mean flow equations.
BC2	Boundary condition parameters for the k-ε equations.
DUMMY1	Scratch array.
FLOW1	Variables dealing with fluid properties and the flow being computed.
GMTRY1	Parameters defining the geometric configuration.
IC1	Variables needed for setting up initial conditions.
IO1	Parameters dealing with program input/output requirements.
METRIC1	Metrics of the nonorthogonal grid transformation, plus the Cartesian coordinates of the grid points.
NUM1	Parameters associated with the numerical method for the mean flow equations.
NUM2	Parameters associated with the numerical method for the k-ε equations.
RSTRT1	Parameters dealing with the restart option.
TIME1	Parameters dealing with the time step selection and convergence determination.
TITLE1	Descriptive title for case being run.
TURB1	Turbulence parameters.
TURB20	Parameters and constants associated with the k-ε equations.

3.2 COMMON VARIABLES LISTED ALPHABETICALLY

In this section all the *Proteus* Fortran variables stored in common blocks are defined, listed alphabetically by variable name. Those marked with an asterisk are input variables. More details on these variables may be found in Section 3.1 of Volume 2. The common block each variable is stored in is given in parentheses at the end of each definition. For subscripted variables, the subscripts are defined along with the variable, except for the subscripts I1, I2, and I3, which are the indices $i, j,$ and k in the ξ, η, and ζ directions, respectively, and run from 1 to N_1, N_2, and N_3.

This list also includes the parameters used as array dimensions. These are not actually stored in a common block, but are stored in the Cray COMDECK PARAMS1. More details may be found in Section 6.2 of Volume 2.

Unless otherwise noted, all variables representing physical quantities are nondimensional. The nondimensionalizing procedure is described in Section 3.1.1 of Volume 2. The type (real or integer) of the variables follows standard Fortran convention, unless stated otherwise. (I.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.)

	Fortran Variable	Symbol	Definition
	A(IV,IS,J,K)	**A**	Subdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
*	APLUS	A^+	Van Driest damping constant in the inner and outer regions of the Baldwin-Lomax turbulence model. (TURB1)
	B(IV,IS,J,K)	**B**	Diagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
	C(IV,IS,J,K)	**C**	Superdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
*	CAVS2E(I)	$\varepsilon_E^{(2)}$ or κ_2	Second order explicit artificial viscosity coefficient in constant coefficient model, or user-specified constant in nonlinear coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	CAVS2I(I)	ε_I	Second order implicit artificial viscosity coefficient in constant coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	CAVS4E(I)	$\varepsilon_E^{(4)}$ or κ_4	Fourth order explicit artificial viscosity coefficient in constant coefficient model, or user-specified constant in nonlinear coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	CB	B	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbu-

			lence model, and in the inner region of the Spalding-Kleinstein turbulence model. (TURB1)
*	CCLAU	K	Clauser constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	CCP	C_{cp}	Constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
	CCP1-4	C_{c_p1} - C_{c_p4}	Constants in empirical formula for specific heat as a function of temperature. (FLOW1)
*	CFL(I)		The ratio $\Delta\tau/\Delta\tau_{cfl}$ where $\Delta\tau$ is the actual time step used in the implicit calculation and $\Delta\tau_{cfl}$ is the allowable time step based on the Courant-Friedrichs-Lewy (CFL) criterion for explicit methods. I is the time step sequence number, and runs from 1 to NTSEQ. (TIME1)
*	CFLMAX		Maximum allowed value of the CFL number. (TIME1)
*	CFLMIN		Minimum allowed value of the CFL number. (TIME1)
	CHGAVG(I)	$\Delta \mathbf{Q}_{avg}$	Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps.[4] The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (TIME1)
	CHGMAX(I,J)	$\Delta \mathbf{Q}_{max}$	Maximum change in absolute value of in the dependent variables over a single time step.[4] The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables, and J = 1 to NITAVG, the number of time steps used in the moving average option for determining convergence. (TIME1)
*	CHG1		Minimum change, in absolute value, that is allowed in any dependent variable before increasing the time step.[4] (TIME1)
*	CHG2		Maximum change, in absolute value, that is allowed in any dependent variable before decreasing the time step.[4] (TIME1)
*	CKLEB	C_{Kleb}	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	CKMIN	$(C_{Kleb})_{min}$	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
	CK1-2	C_{k1} - C_{k2}	Constants in empirical formula for thermal conductivity coefficient as a function of temperature. (FLOW1)
*	CMUR	C_{μ_r}	Constant used to compute C_μ in the turbulent viscosity formula for the k-ε equations. (TURB20)
	CMU1-2	$C_{\mu1}$ - $C_{\mu2}$	Constants in empirical formula for laminar viscosity coefficient as a function of temperature. (FLOW1)

4 For the energy equation, the change in E_T is divided by $E_{T_r} = \rho_r \overline{R} T_r/(\gamma_r - 1) + u_r^2/2$, so that it is the same order of magnitude as the other conservation variables.

*	CNL	n	Exponent in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model. (TURB1)
*	CONE	C_1	Constant used in the production term of the ε equation. (TURB20)
	CP(I1,I2,I3)	c_p	Specific heat at constant pressure at time level n. (FLOW1)
*	CTHREE	C_3	Constant used to compute C_μ in the turbulent viscosity formula for the k-ε equations. (TURB20)
*	CTWOR	C_{2_r}	Constant used to compute C_2 in the dissipation term of the ε equation. (TURB20)
	CV(I1,I2,I3)	c_v	Specific heat at constant volume at time level n. (FLOW1)
*	CVK	κ	Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein turbulence models. (TURB1)
*	CWK	C_{wk}	Constant used in the formula for F_{wake} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
	DEL	$\Delta\xi$, $\Delta\eta$, or $\Delta\zeta$	Computational grid spacing in the ADI sweep direction. (NUM1)
	DETA	$\Delta\eta$	Computational grid spacing in the η direction. (NUM1)
	DPDET(I)	$\partial p/\partial E_T$	The derivative of p with respect to E_T, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DPDRHO(I)	$\partial p/\partial \rho$	The derivative of p with respect to ρ, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DPDRU(I)	$\partial p/\partial(\rho u)$	The derivative of p with respect to ρu, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DPDRV(I)	$\partial p/\partial(\rho v)$	The derivative of p with respect to ρv, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DPDRW(I)	$\partial p/\partial(\rho w)$	The derivative of p with respect to ρw, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
*	DT(I)	Δt	The time step size, when specified directly as input. I is the time step sequence number, and runs from 1 to NTSEQ. (TIME1)
	DTAU(I1,I2,I3)	$\Delta\tau$	Computational time step size. (TIME1)
	DTDET(I)	$\partial T/\partial E_T$	The derivative of T with respect to E_T, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

	DTDRHO(I)	$\partial T/\partial\rho$	The derivative of T with respect to ρ, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DTDRU(I)	$\partial T/\partial(\rho u)$	The derivative of T with respect to ρu, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DTDRV(I)	$\partial T/\partial(\rho v)$	The derivative of T with respect to ρv, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	DTDRW(I)	$\partial T/\partial(\rho w)$	The derivative of T with respect to ρw, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
*	DTF1		Factor by which the time step is multiplied if the solution changes too slowly. (TIME1)
*	DTF2		Factor by which the time step is divided if the solution changes too quickly. (TIME1)
*	DTMAX		Maximum value that $\Delta\tau$ is allowed to reach, or the maximum $\Delta\tau$ used in the time step cycling procedure. (TIME1)
*	DTMIN		Minimum value that $\Delta\tau$ is allowed to reach, or the minimum $\Delta\tau$ used in the time step cycling procedure. (TIME1)
	DUMMY(I1,I2,I3)		Dummy array used for temporary storage in several subroutines. (DUMMY1)
	DW(I1,I2,I3,I)	$\Delta\hat{\mathbf{W}}^n$ or $\Delta\hat{\mathbf{W}}^*$	Unknown vector in the LU solution of the k-ε equations. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively. (NUM2)
	DXI	$\Delta\xi$	Computational grid spacing in the ξ direction. (NUM1)
	DZETA	$\Delta\zeta$	Computational grid spacing in the ζ direction. (NUM1)
	E(I1,I2,I3)	ε	Turbulent dissipation rate at time level n. (TURB20)
	EL(I1,I2,I3)	ε	Turbulent dissipation rate at previous or intermediate time level. (TURB20)
*	EPS(I)	ε	Convergence level to be reached. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (TIME1)
	EP1-2		Parameters used in various parts of the code as minimum and maximum allowed values. (PARAMS1)
	ER	e_r	Dimensional reference energy, $\rho_r u_r^2$. (FLOW1)
	ET(I1,I2,I3)	E_T	Total energy at time level n. (FLOW1)
	ETAT(I1,I2,I3)	η_t	The derivative of the computational coordinate η with respect to untransformed time t. (METRIC1)

ETAX(I1,I2,I3)	η_x	The derivative of the computational coordinate η with respect to the Cartesian coordinate x. (METRIC1)
ETAY(I1,I2,I3)	η_y	The derivative of the computational coordinate η with respect to the Cartesian coordinate y. (METRIC1)
ETAZ(I1,I2,I3)	η_z	The derivative of the computational coordinate η with respect to the Cartesian coordinate z. (METRIC1)
ETL(I1,I2,I3)	E_T	Total energy at previous or intermediate time level. (FLOW1)
* FBCT1(I2,I3,I,J)		Point-by-point values used for boundary conditions for the k-ε turbulence model on the $\xi = 0$ and $\xi = 1$ boundaries. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC2)
* FBCT2(I1,I3,I,J)		Point-by-point values used for boundary conditions for the k-ε turbulence model on the $\eta = 0$ and $\eta = 1$ boundaries. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)
* FBCT3(I1,I2,I,J)		Point-by-point values used for boundary conditions for the k-ε turbulence model on the $\zeta = 0$ and $\zeta = 1$ boundaries. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC2)
* FBC1(I2,I3,I,J)		Point-by-point values used for steady boundary conditions on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)
* FBC2(I1,I3,I,J)		Point-by-point values used for steady boundary conditions on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
* FBC3(I1,I2,I,J)		Point-by-point values used for steady boundary conditions on the $\zeta = 0$ and $\zeta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
* FPMIN		Value used to cut off the search for F_{max} in the outer region part of the Baldwin-Lomax turbulence model. (TURB1)

*	GAMR	γ_r	Reference ratio of specific heats, c_{p_r}/c_{v_r}. (FLOW1)
*	GBCT1(I,J)		Values used for boundary conditions for the k-ε turbulence model on the $\xi = 0$ and $\xi = 1$ boundaries, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC2)
*	GBCT2(I,J)		Values used for boundary conditions for the k-ε turbulence model on the $\eta = 0$ and $\eta = 1$ boundaries, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)
*	GBCT3(I,J)		Values used for boundary conditions for the k-ε turbulence model on the $\zeta = 0$ and $\zeta = 1$ boundaries, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC2)
*	GBC1(I,J)		Values used for steady boundary conditions on the $\xi = 0$ and $\xi = 1$ boundaries, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)
*	GBC2(I,J)		Values used for steady boundary conditions on the $\eta = 0$ and $\eta = 1$ boundaries, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
*	GBC3(I,J)		Values used for steady boundary conditions on the $\zeta = 0$ and $\zeta = 1$ boundaries, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
	GC	g_c	Dimensional proportionality factor in Newton's second law, either 32.174 lb_m-ft/lb_f-sec^2, or 1.0 kg-m/N-sec^2. (FLOW1)
*	GTBC1(K,I,J)		A variable used to specify the values for unsteady and time-periodic boundary conditions on the $\xi = 0$ and $\xi = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. For general unsteady boundary conditions, K = 1 to NTBC, corresponding to the time steps in the array NTBCA, and GTBC1 specifies the boundary condition value directly. For time-periodic boundary conditions, K = 1 to 4, and GTBC1 specifies the four coefficients in the equation used to determine the boundary condition value. (BC1)
*	GTBC2(K,I,J)		A variable used to specify the values for unsteady and time-periodic boundary conditions on the $\eta = 0$ and $\eta = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq}

		conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. For general unsteady boundary conditions, K = 1 to NTBC, corresponding to the time steps in the array NTBCA, and GTBC2 specifies the boundary condition value directly. For time-periodic boundary conditions, K = 1 to 4, and GTBC2 specifies the four coefficients in the equation used to determine the boundary condition value. (BC1)
* GTBC3(K,I,J)		A variable used to specify the values for unsteady and time-periodic boundary conditions on the $\zeta = 0$ and $\zeta = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. For general unsteady boundary conditions, K = 1 to NTBC, corresponding to the time steps in the array NTBCA, and GTBC3 specifies the boundary condition value directly. For time-periodic boundary conditions, K = 1 to 4, and GTBC3 specifies the four coefficients in the equation used to determine the boundary condition value. (BC1)
HSTAG	h_T	Stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)
* HSTAGR	h_{T_r}	Dimensional stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)
* IAV2E		Flag for second-order explicit artificial viscosity; 0 for none, 1 for constant coefficient model, 2 for nonlinear coefficient model. (NUM1)
* IAV2I		Flag for second-order implicit artificial viscosity; 0 for none, 1 for constant coefficient model. (NUM1)
* IAV4E		Flag for fourth-order explicit artificial viscosity; 0 for none, 1 for constant coefficient model, 2 for nonlinear coefficient model. (NUM1)
IBASE		Base index used with ISTEP to compute one-dimensional index for three-dimensional array. Then, for example, for any sweep U(I1,I2,I3) = U1(IBASE + ISTEP*(I − 1)) where I is the grid index in the sweep direction. (NUM1)
IBCELM(I,J)		Flags for elimination of off-diagonal sub-matrices resulting from gradient or extrapolation boundary conditions: 0 if elimination is not necessary, 1 if it is. The subscript I = 1, 2, or 3 corresponding to the sweep direction, and J = 1 or 2 corresponding to the lower or upper boundary in that direction. (BC1)
* IBCT1(I2,I3,I,J)		Flags specifying, point-by-point, the type of boundary conditions used for the k-ε turbulence model on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC2)

* IBCT2(I1,I3,I,J)	Flags specifying, point-by-point, the type of boundary conditions used for the k-ε turbulence model on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)
* IBCT3(I1,I2,I,J)	Flags specifying, point-by-point, the type of boundary conditions used for the k-ε turbulence model on the $\zeta = 0$ and $\zeta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC2)
* IBC1(I2,I3,I,J)	Flags specifying, point-by-point, the type of steady boundary conditions used on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)
* IBC2(I1,I3,I,J)	Flags specifying, point-by-point, the type of steady boundary conditions used on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
* IBC3(I1,I2,I,J)	Flags specifying, point-by-point, the type of steady boundary conditions used on the $\zeta = 0$ and $\zeta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
IBVUP(I,J)	Flags for updating boundary values from the first two sweeps after the last sweep: 0 if updating is not necessary, 1 if it is. Updating is required when gradient or extrapolation boundary conditions are used. The subscript I = 1 or 2 corresponding to the sweep direction, and J = 1 or 2 corresponding to the lower or upper boundary in that direction. (BC1)
* ICHECK	Results are checked for convergence every ICHECK'th time level. (TIME1)
ICONV	Convergence flag; 0 if not converged, 1 if converged. (TIME1)
* ICTEST	Flag for convergence criteria to be used. (TIME1)
* ICVARS	Parameter specifying which variables are being supplied as initial conditions by subroutine INIT. (FLOW1)

*	IDEBUG(I)	A 20-element array of flags specifying various debug options. (IO1)
*	IDTAU	Flag for time step selection method. (TIME1)
*	IDTMOD	The time step size is modified every IDTMOD'th time step. (TIME1)
*	IEULER	Flag for Euler calculation option; 0 for a full time-averaged Navier-Stokes calculation, 1 for an Euler calculation. (FLOW1)
	IGAM	Flag set by method used to select GAMR; 0 if GAMR is defaulted (and hence c_p and c_v are functions of temperature), 1 if GAMR is specified by user (and hence c_p and c_v are constants). (FLOW1)
	IGINT(I)	Flags for grid interpolation requirement; 0 if interpolation is not needed, 1 if interpolation is needed. The subscript I = 1 to 3, corresponding to the ξ, η, and ζ directions, respectively. (GMTRY1)
*	IHSTAG	Flag for constant stagnation enthalpy option; 0 to solve the energy equation, 1 to eliminate the energy equation by assuming constant stagnation enthalpy. (FLOW1)
*	ILAMV	Flag for computation of laminar viscosity and thermal conductivity; 0 for constant values, 1 for functions of local temperature. (FLOW1)
*	ILDAMP	Flag for the Launder-Priddin modified mixing length formula in the inner region of the Baldwin-Lomax turbulence model. (TURB1)
	INEG	Flag indicating non-positive values of pressure and/or temperature: 0 for no non-positive values, 1 for some. (FLOW1)
*	INNER	Flag for type of inner region turbulence model. (TURB1)
*	IPACK(I)	Flags for grid packing option; 0 for no packing, 1 to pack points as specified by the input array SQ. The subscript I = 1 to 3, corresponding to the ξ, η, and ζ directions, respectively. (NUM1)
*	IPLOT	Flag controlling the creation of an auxiliary file, usually called a "plot file", used for later post-processing. (IO1)
*	IPLT	Results are written into the plot file every IPLT time steps. (IO1)
*	IPLTA(I)	Time levels at which results are written into the plot file. The subscript I = 1 to 101, the maximum number of time levels that may be written. (IO1)
*	IPRT	Results are printed every IPRT time levels. (IO1)
*	IPRTA(I)	Time levels at which results are printed. The subscript I = 1 to 101, the maximum number of time levels that may be printed. (IO1)

*	IPRT1		Results are printed at every IPRT1'th mesh point in the ξ direction. (IO1)
*	IPRT2		Results are printed at every IPRT2'th mesh point in the η direction. (IO1)
*	IPRT3		Results are printed at every IPRT3'th mesh point in the ζ direction. (IO1)
*	IPRT1A(I)		ξ indices at which results are printed. The subscript I = 1 to a maximum of N1, the number of grid points in the ξ direction. (IO1)
*	IPRT2A(I)		η indices at which results are printed. The subscript I = 1 to a maximum of N2, the number of grid points in the η direction. (IO1)
*	IPRT3A(I)		ζ indices at which results are printed. The subscript I = 1 to a maximum of N3, the number of grid points in the ζ direction. (IO1)
*	IREST		Flag controlling the reading and writing of auxiliary files used for restarting the calculation in a separate run. (RSTRT1)
	ISTEP		Multiplication factor used with IBASE to compute one-dimensional index for three-dimensional array. (NUM1)
	ISWEEP		Flag specifying ADI sweep direction; 1 for ξ direction, 2 for η direction, and 3 for ζ direction. (NUM1)
	IT	n	Current time step number, or known time level. Time step number n updates the solution from time level n to $n+1$. (TIME1)
	ITBEG		The time time step number, or known time level n, at the beginning of a run. For a non-restart case, ITBEG = 1. (TIME1)
	ITDBC		Flag for time-dependent boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used. (BC1)
	ITEND		The final time step number. (TIME1)
*	ITHIN(I)		Flag for thin layer option; 0 to include second derivative viscous terms, 1 to eliminate them. The subscript I = 1 to 3, corresponding to the ξ, η, and ζ directions, respectively. (FLOW1)
	ITSEQ		Current time step sequence number. (TIME1)
*	ITURB		Flag for turbulent flow option; 0 for laminar flow, 1 for turbulent flow using the Baldwin-Lomax algebraic turbulence model, 20 for turbulent flow using the Chien two-equation k-ε turbulence model. (TURB1)
*	IUNITS		Flag for type of units used to specify reference conditions; 0 for English units, 1 for SI units. (IO1)

IV i_v Grid point index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, $\overline{IV} = j$ for the first sweep and i for the second and third sweeps. (NUM1)

* IVOUT(I) A 50-element array specifying which variables are to be printed. (IO1)

* IWALL1(I) Flags indicating type of surfaces in the ξ direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ surfaces, respectively. (TURB1)

* IWALL2(I) Flags indicating type of surfaces in the η direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ surfaces, respectively. (TURB1)

* IWALL3(I) Flags indicating type of surfaces in the ζ direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ surfaces, respectively. (TURB1)

* IWOUT1(I) Flags specifying whether or not various parameters are to be printed along the ξ boundaries; 0 for no printout, 1 for printout along the boundary. The subscript I = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (IO1)

* IWOUT2(I) Flags specifying whether or not various parameters are to be printed along the η boundaries; 0 for no printout, 1 for printout along the boundary. The subscript I = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (IO1)

* IWOUT3(I) Flags specifying whether or not various parameters are to be printed along the ζ boundaries; 0 for no printout, 1 for printout along the boundary. The subscript I = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (IO1)

I1 i Grid point index in the ξ direction. (NUM1)

I2 j Grid point index in the η direction. (NUM1)

I3 k Grid point index in the ζ direction. (NUM1)

* JBCT1(I,J) Flags specifying the type of boundary conditions used for the k-ε turbulence model on the $\xi = 0$ and $\xi = 1$ surfaces, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC2)

* JBCT2(I,J) Flags specifying the type of boundary conditions used for the k-ε turbulence model on the $\eta = 0$ and $\eta = 1$ surfaces, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1

or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)

* JBCT3(I,J)		Flags specifying the type of boundary conditions used for the k-ε turbulence model on the $\zeta = 0$ and $\zeta = 1$ surfaces, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC2)
* JBC1(I,J)		Flags specifying the type of steady boundary conditions used on the $\xi = 0$ and $\xi = 1$ surfaces, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)
* JBC2(I,J)		Flags specifying the type of steady boundary conditions used on the $\eta = 0$ and $\eta = 1$ surfaces, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
* JBC3(I,J)		Flags specifying the type of steady boundary conditions used on the $\zeta = 0$ and $\zeta = 1$ surfaces, when specified for the entire surface. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
JI(I1,I2,I3)	J^{-1}	Inverse Jacobian of the non-orthogonal grid transformation. This is a real variable. (METRIC1)
* JTBC1(I,J)		A variable specifying the type of time dependency for the boundary conditions on the $\xi = 0$ and $\xi = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)
* JTBC2(I,J)		A variable specifying the type of time dependency for the boundary conditions on the $\eta = 0$ and $\eta = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
* JTBC3(I,J)		A variable specifying the type of time dependency for the boundary conditions on the $\zeta = 0$ and $\zeta = 1$ boundaries. I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
KBCPER(I)		Flags for spatially periodic boundary conditions: 0 for non-periodic, 1 for periodic. The subscript I = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. (BC1)
* KBC1(J)		Flags for type of boundaries in the ξ direction. The subscript J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)

*	KBC2(J)		Flags for type of boundaries in the η direction. The subscript $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)
*	KBC3(J)		Flags for type of boundaries in the ζ direction. The subscript $J = 1$ or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (BC1)
	KE(I1,I2,I3)	k	Turbulent kinetic energy at time level n. This is a real variable. (TURB20)
	KEL(I1,I2,I3)	k	Turbulent kinetic energy at previous or intermediate time level. This is a real variable. (TURB20)
	KT(I1,I2,I3)	k	Effective thermal conductivity coefficient at time level n. This is a real variable. (FLOW1)
*	KTR	k_r	Dimensional reference thermal conductivity coefficient. This is a real variable. (FLOW1)
	LA(I1,I2,I3)	λ	Effective second coefficient of viscosity at time level n (usually assumed equal to $-2\mu/3$.) This is a real variable. (FLOW1)
*	LR	L_r	Dimensional reference length. This is a real variable. (FLOW1)
	LRMAX(I,J,K)		The grid indices corresponding to the location of the maximum absolute value of the residual. The subscript $I = 1$ to 3, corresponding to the ξ, η, and ζ directions, respectively, $J = 1$ to N_{eq}, corresponding to the N_{eq} coupled governing equations, and $K = 1$ or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
*	LWALL1(I2,I3,I)		Flags indicating, point-by-point, the type of surfaces in the ξ direction; 0 for a free boundary, 1 for a solid wall. The subscript $I = 1$ or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (TURB1)
*	LWALL2(I1,I3,I)		Flags indicating, point-by-point, the type of surfaces in the η direction; 0 for a free boundary, 1 for a solid wall. The subscript $I = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (TURB1)
*	LWALL3(I1,I2,I)		Flags indicating, point-by-point, the type of surfaces in the ζ direction; 0 for a free boundary, 1 for a solid wall. The subscript $I = 1$ or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. (TURB1)
	LWSET(I)		Flags specifying how wall locations are determined for the turbulence model; 0 if wall locations are found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters. The subscript $I = 1$ to 6, corresponding to the $\xi = 0$, $\xi = 1$, $\eta = 0$, $\eta = 1$, $\zeta = 0$, and $\zeta = 1$ boundaries, respectively. (TURB1)
*	MACHR	M_r	Reference Mach number, $u_r/(\gamma_r \bar{R} T_r)^{1/2}$. This is a real variable. (FLOW1)

METT(IV,IS)	ξ_t, η_t, or ζ_t	The derivative of the computational coordinate in the ADI sweep direction with respect to untransformed time t. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
METX(IV,IS)	ξ_x, η_x, or ζ_x	The derivative of the computational coordinate in the ADI sweep direction with respect to the Cartesian coordinate x. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
METY(IV,IS)	ξ_y, η_y, or ζ_y	The derivative of the computational coordinate in the ADI sweep direction with respect to the Cartesian coordinate y. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
METZ(IV,IS)	ξ_z, η_z, or ζ_z	The derivative of the computational coordinate in the ADI sweep direction with respect to the Cartesian coordinate z. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
MU(I1,I2,I3)	μ	Effective viscosity coefficient at time level n. This is a real variable. (FLOW1)
* MUR	μ_r	Dimensional reference viscosity coefficient. This is a real variable. (FLOW1)
MUT(I1,I2,I3)	μ_T	Turbulent viscosity coefficient at time level n. This is a real variable. (FLOW1)
MUTL(I1,I2,I3)	μ_t	Turbulent viscosity coefficient at previous or intermediate time level. This is a real variable. (TURB20)
NAMAX		A dimensioning parameter equal to the maximum number of time steps allowed in the moving average convergence test (the ICTEST = 2 option). (PARAMS1)
NBC		A dimensioning parameter equal to the number of boundary conditions per equation. (PARAMS1)
NC		Array index associated with the continuity equation. (NUM1)
NDIAGP		Number of diagonals containing interior points in a ξ-η plane. (PARAMS1)
* NDTCYC		Number of time steps per cycle used in the time step cycling procedure. (TIME1)

	NEN		Array index associated with the energy equation. (NUM1)
	NEQ	N_{eq}	The number of coupled governing equations actually being solved. (NUM1)
	NEQP		A dimensioning parameter equal to the number of coupled equations allowed. (PARAMS1)
	NEQPM		A dimensioning parameter equal to the maximum number of coupled equations available. (PARAMS1)
	NET		Array index associated with the dependent variable E_T. (NUM1)
*	NGEOM		Flag used to specify type of computational coordinates; 1 for Cartesian (x,y,z) coordinates, 2 for cylindrical (r,θ,x) coordinates, and 10 to read the coordinates from unit NGRID. (GMTRY1)
*	NGRID		Unit number for reading grid file. (IO1)
*	NHIST		Unit number for writing convergence history file. (IO1)
*	NHMAX		Maximum number of time levels allowed in the printout of the convergence history file (not counting the first two, which are always printed.) (IO1)
	NIN		Unit number for reading namelist input. (IO1)
*	NITAVG		Number of time steps used in the moving average convergence test. (TIME1)
	NMAXP		A dimensioning parameter equal to the maximum of N1P, N2P, and N3P. (PARAMS1)
*	NOUT		Unit number for writing standard output. (IO1)
*	NPLOT		Unit number for writing CONTOUR or PLOT3D Q plot file. (IO1)
*	NPLOTX		Unit number for writing PLOT3D XYZ plot file. (IO1)
	NPNTP		Number of interior points in a ξ-η plane. (PARAMS1)
	NPRT1		Total number of indices for printout in the ξ direction. (IO1)
	NPRT2		Total number of indices for printout in the η direction. (IO1)
	NPRT3		Total number of indices for printout in the ζ direction. (IO1)
	NPTS	N	The number of grid points in the sweep direction. (NUM1)
	NPT1	N_1 or $N_1 + 1$	The number of grid points in the ξ direction used in computing coefficients: N_1 for non-periodic boundary conditions; $N_1 + 1$ for spatially periodic boundary conditions. (NUM1)
	NPT2	N_2 or $N_2 + 1$	The number of grid points in the η direction used in computing coefficients: N_2 for non-periodic boundary conditions; $N_2 + 1$ for spatially periodic boundary conditions. (NUM1)

NPT3	N_3 or $N_3 + 1$	The number of grid points in the ζ direction used in computing coefficients: N_3 for non-periodic boundary conditions; $N_3 + 1$ for spatially periodic boundary conditions. (NUM1)
NR		Array index associated with the dependent variable ρ. (NUM1)
* NRQIN		Unit number for reading restart flow field. (RSTRT1)
* NRQOUT		Unit number for writing restart flow field. (RSTRT1)
NRU		Array index associated with the dependent variable ρu. (NUM1)
NRV		Array index associated with the dependent variable ρv. (NUM1)
NRW		Array index associated with the dependent variable ρw. (NUM1)
* NRXIN		Unit number for reading restart computational mesh. (RSTRT1)
* NRXOUT		Unit number for writing restart computational mesh. (RSTRT1)
* NTBC		Number of values in the tables of GTBC1, GTBC2, and/or GTBC3 vs. NTBCA for general unsteady boundary conditions. (BC1)
* NTBCA(I)		Time step values at which GTBC1, GTBC2, and/or GTBC3 are specified for general unsteady boundary conditions. The subscript I = 1 to NTBC, corresponding to the NTBC values in the table. (BC1)
* NTIME(I)		Maximum number of time steps to march. I runs from 1 to NTSEQP, corresponding to the time step sequence number. (TIME1)
* NTKE		Number of k-ε iterations per mean flow iteration. (TURB20)
NTOTP		A dimensioning parameter equal to the total storage required for a single three-dimensional array (i.e., N1P × N2P × N3P). (PARAMS1)
NTP		A dimensioning parameter equal to the maximum number of entries in the table of time-dependent boundary condition values. (PARAMS1)
* NTSEQ		The total number of time step sequences being used. (TIME1)
NTSEQP		A dimensioning parameter equal to the maximum number of time step sequences in the time step sequencing option. (PARAMS1)
NV	N_v	The number of grid points in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are

			vectorized). Therefore, NV = N_2 for the first sweep and N_1 for the second and third sweeps. (NUM1)
	NXM		Array index associated with the x-momentum equation. (NUM1)
	NYM		Array index associated with the y-momentum equation. (NUM1)
	NZM		Array index associated with the z-momentum equation. (NUM1)
*	N1	N_1	The number of grid points in the ξ direction. (NUM1)
	N1P		A dimensioning parameter equal to the maximum number of grid points in the ξ direction. (PARAMS1)
*	N2	N_2	The number of grid points in the η direction. (NUM1)
	N2P		A dimensioning parameter equal to the maximum number of grid points in the η direction. (PARAMS1)
*	N3	N_3	The number of grid points in the ζ direction. (NUM1)
	N3P		A dimensioning parameter equal to the maximum number of grid points in the ζ direction. (PARAMS1)
	P(I1,I2,I3)	p	Static pressure at time level n. (FLOW1)
	PR	p_r	Dimensional reference static pressure, $\rho_r \bar{R} T_r / g_c$. (FLOW1)
*	PRLR	Pr_{l_r}	Reference laminar Prandtl number, $c_{p_r}\mu_r/k_r$, where $c_{p_r} = \gamma_r \bar{R}/(\gamma_r - 1)$. (FLOW1)
	PRR	Pr_r	Reference Prandtl number, $\mu_r u_r^2 / k_r T_r$. (FLOW1)
*	PRT	Pr_t	Turbulent Prandtl number, or, if non-positive, a flag indicating the use of a variable turbulent Prandtl number. (TURB1)
*	P0	p_0	Initial static pressure. (IC1)
*	RER	Re_r	Reference Reynolds number, $\rho_r u_r L_r / \mu_r$. (FLOW1)
	RESAVG(J,K)	R_{avg}	The average absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
	RESL2(J,K)	R_{L_2}	The L_2 norm of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
	RESMAX(J,K)	R_{max}	The maximum absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corre-

			sponding to the residual computed without and with the artificial viscosity terms. (TIME1)
*	RG	$\overline{R}$	Dimensional gas constant. (FLOW1)
	RGAS	R	Non-dimensional gas constant. (FLOW1)
	RHO(I1,I2,I3)	ρ	Static density at time level n. (FLOW1)
	RHOL(I1,I2,I3)	ρ	Static density at previous or intermediate time level. (FLOW1)
*	RHOR	ρ_r	Dimensional reference density. (FLOW1)
*	RMAX	r_{max}	Maximum r coordinate for cylindrical grid option. (GMTRY1)
*	RMIN	r_{min}	Minimum r coordinate for cylindrical grid option. (GMTRY1)
	S(IV,IS,J)	**S**	Subvector of source terms in the block tridiagonal system of equations. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript $J = 1$ to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	SIGE	σ_ε	Constant used in the diffusion term of the ε equation. (TURB20)
*	SIGK	σ_k	Constant used in the diffusion term of the k equation. (TURB20)
*	SQ(I,J)		An array controlling the packing of grid points using the Roberts transformation. The subscript $I = 1$ to 3, corresponding to the ξ, η, and ζ directions, respectively. SQ(I,1) specifies the location of packing, and SQ(I,2) specifies the amount of packing. (NUM1)
	T(I1,I2,I3)	T	Static temperature at time level n. (FLOW1)
	TAU(I1,I2,I3)	τ	Current value of the time marching parameter. (TIME1)
*	TFACT		Factor used in computing the k-ε time step, $\Delta\tau_{k\text{-}\varepsilon} = \text{TFACT}(\Delta\tau)$. (TURB20)
*	THC(I)	$\theta_1,\ \theta_2$	A two-element array specifying the time difference centering parameters used for the continuity equation. (NUM1)
*	THE(I)	$\theta_1,\ \theta_2,\ \theta_3$	A three-element array specifying the time difference centering parameters used for the energy equation. (NUM1)
*	THKE(I)	$\theta_1,\ \theta_2$	A two-element array specifying the time difference centering parameters used for the k-ε equations. (NUM2)
*	THMAX	θ_{max}	Maximum θ coordinate in degrees for cylindrical grid option. (GMTRY1)

* THMIN	θ_{min}	Minimum θ coordinate in degrees for cylindrical grid option. (GMTRY1)
* THX(I)	$\theta_1,\ \theta_2,\ \theta_3$	A three-element array specifying the time difference centering parameters used for the x-momentum equation. (NUM1)
* THY(I)	$\theta_1,\ \theta_2,\ \theta_3$	A three-element array specifying the time difference centering parameters used for the y-momentum equation. (NUM1)
* THZ(I)	$\theta_1,\ \theta_2,\ \theta_3$	A three-element array specifying the time difference centering parameters used for the z-momentum equation. (NUM1)
* TITLE		Title for printed output and CONTOUR plot file, up to 72 characters long. This is a character variable. (TITLE1)
TL(I1,I2,I3)	T	Static temperature at previous or intermediate time level. (FLOW1)
* TLIM		When the amount of CPU time remaining for the job drops below TLIM seconds, the calculation is stopped. (TIME1)
* TR	T_r	Dimensional reference temperature. (FLOW1)
* T0	T_0	Initial static temperature. (IC1)
U(I1,I2,I3)	u	Velocity in the Cartesian x direction at time level n. (FLOW1)
UL(I1,I2,I3)	u	Velocity in the Cartesian x direction at previous or intermediate time level. (FLOW1)
* UR	u_r	Dimensional reference velocity. (FLOW1)
* U0	u_0	Initial velocity in the Cartesian x direction. (IC1)
V(I1,I2,I3)	v	Velocity in the Cartesian y direction at time level n. (FLOW1)
VL(I1,I2,I3)	v	Velocity in the Cartesian y direction at previous or intermediate time level. (FLOW1)
VORT(I1,I2,I3)	$\lvert\vec{\Omega}\rvert$	Total vorticity magnitude. (TURB1)
VORT(I1,I2,I3)	P_k	Production rate of turbulent kinetic energy. (TURB1)
* V0	v_0	Initial velocity in the Cartesian y direction. (IC1)
W(I1,I2,I3)	w	Velocity in the Cartesian z direction at time level n. (FLOW1)
WL(I1,I2,I3)	w	Velocity in the Cartesian z direction at previous or intermediate time level. (FLOW1)
* W0	w_0	Initial velocity in the Cartesian z direction. (IC1)
X(I1,I2,I3)	x	Cartesian x coordinate. (METRIC1)
XIT(I1,I2,I3)	ξ_t	The derivative of the computational coordinate ξ with respect to untransformed time t. (METRIC1)

	XIX(I1,I2,I3)	ξ_x	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate x. (METRIC1)
	XIY(I1,I2,I3)	ξ_y	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate y. (METRIC1)
	XIZ(I1,I2,I3)	ξ_z	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate z. (METRIC1)
*	XMAX	x_{max}	Maximum x coordinate for Cartesian or cylindrical grid option. (GMTRY1)
*	XMIN	x_{min}	Minimum x coordinate for Cartesian or cylindrical grid option. (GMTRY1)
	Y(I1,I2,I3)	y	Cartesian y coordinate. (METRIC1)
*	YMAX	y_{max}	Maximum y coordinate for Cartesian grid option. (GMTRY1)
*	YMIN	y_{min}	Minimum y coordinate for Cartesian grid option. (GMTRY1)
	YPLUSD(I1,I2,I3)	y^+	Non-dimensional distance from the nearest solid wall. (TURB20)
	Z(I1,I2,I3)	z	Cartesian z coordinate. (METRIC1)
	ZETAT(I1,I2,I3)	ζ_t	The derivative of the computational coordinate ζ with respect to untransformed time t. (METRIC1)
	ZETAX(I1,I2,I3)	ζ_x	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate x. (METRIC1)
	ZETAY(I1,I2,I3)	ζ_y	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate y. (METRIC1)
	ZETAZ(I1,I2,I3)	ζ_z	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate z. (METRIC1)
*	ZMAX	z_{max}	Maximum z coordinate for Cartesian grid option. (GMTRY1)
*	ZMIN	z_{min}	Minimum z coordinate for Cartesian grid option. (GMTRY1)

3.3 COMMON VARIABLES LISTED SYMBOLICALLY

In this section many of the *Proteus* Fortran variables stored in common blocks are defined, listed symbolically. Note that this list does not include those variables without symbolic representations, such as various flags, or those whose meaning depends on other parameters, such as the boundary condition values and sweep direction metrics. The variables marked with an asterisk are input variables. More details on these may be found in Section 3.1 of Volume 2. The common block each variable is stored in is given in parentheses at the end of each definition. For subscripted variables, the subscripts are defined along with the variable, except for the subscripts I1, I2, and I3, which are the indices i, j, and k in the ξ, η, and ζ directions, respectively, and run from 1 to N_1, N_2, and N_3.

Unless otherwise noted, all variables representing physical quantities are nondimensional. The nondimensionalizing procedure is described in Section 3.1.1 of Volume 2. The type (real or integer) of the variables follows standard Fortran convention, unless stated otherwise. (I.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.)

	Symbol	Fortran Variable	Definition
*	A^+	APLUS	Van Driest damping constant in the inner and outer regions of the Baldwin-Lomax turbulence model. (TURB1)
	A	AMAT(IV,IS,J,K)	Subdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
*	B	CB	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbulence model, and in the inner region of the Spalding-Kleinstein turbulence model. (TURB1)
	B	BMAT(IV,IS,J,K)	Diagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
	c_p	CP(I1,I2,I3)	Specific heat at constant pressure at time level n. (FLOW1)
	c_v	CV(I1,I2,I3)	Specific heat at constant volume at time level n. (FLOW1)
*	C_{cp}	CCP	Constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
	C_{c_p1} - C_{c_p4}	CCP1-CCP4	Constants in empirical formula for specific heat as a function of temperature. (FLOW1)
	C_{k1} - C_{k2}	CK1-2	Constants in empirical formula for thermal conductivity coefficient as a function of temperature.
*	C_{Kleb}	CKLEB	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	$(C_{Kleb})_{min}$	CKMIN	Constant used in the formula for the Klebanoff intermittency factor F_{Kleb} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	C_{μ_r}	CMUR	Constant used to compute C_μ in the turbulent viscosity formula for the k-ε equations. (TURB20)

	$C_{\mu 1}$ - $C_{\mu 2}$	CMU1-2	Constants in empirical formula for laminar viscosity coefficient as a function of temperature. (FLOW1)
*	C_{wk}	CWK	Constant used in the formula for F_{wake} in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	C_1	CONE	Constant used in the production term of the ε equation. (TURB20)
*	C_{2_r}	CTWOR	Constant used to compute C_2 in the dissipation term of the ε equation. (TURB20)
*	C_3	CTHREE	Constant used to compute C_μ in the turbulent viscosity formula for the k-ε equations. (TURB20)
	$\mathbf{C}$	CMAT(IV,IS,J,K)	Superdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (NUM1)
	e_r	ER	Dimensional reference energy, $\rho_r u_r^2$. (FLOW1)
	E_T	ET(I1,I2,I3)	Total energy at time level n. (FLOW1)
	E_T	ETL(I1,I2,I3)	Total energy at previous or intermediate time level. (FLOW1)
	g_c	GC	Dimensional proportionality factor in Newton's second law, either 32.174 lb_m-ft/lb_f-sec^2, or 1.0 kg-m/N-sec^2. (FLOW1)
	h_T	HSTAG	Constant stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)
*	h_{T_r}	HSTAGR	Dimensional stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)
	i	I1	Grid point index in the ξ direction. (NUM1)
	i_v	IV	Grid point index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, IV = j for the first sweep and i for the second and third sweeps. (NUM1)
	j	I2	Grid point index in the η direction. (NUM1)
	J^{-1}	JI(I1,I2,I3)	Inverse Jacobian of the non-orthogonal grid transformation. This is a real variable. (METRIC1)
	k	I3	Grid point index in the ζ direction. (NUM1)
	k	KT(I1,I2,I3)	Effective thermal conductivity coefficient at time level n. This is a real variable. (FLOW1)
	k	KE(I1,I2,I3)	Turbulent kinetic energy at time level n. This is a real variable. (TURB20)

	k	KEL(I1,I2,I3)	Turbulent kinetic energy at previous or intermediate time level. This is a real variable. (TURB20)
*	k_r	KTR	Dimensional reference thermal conductivity coefficient. This is a real variable. (FLOW1)
*	K	CCLAU	Clauser constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)
*	L_r	LR	Dimensional reference length. This is a real variable. (FLOW1)
*	M_r	MACHR	Reference Mach number, $u_r/(\gamma_r \bar{R} T_r)^{1/2}$. This is a real variable. (FLOW1)
	n	IT	Current time step number, or known time level. Time step number n updates the solution from time level n to $n+1$. (TIME1)
*	n	CNL	Exponent in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model. (TURB1)
	N	NPTS	The number of grid points in the sweep direction. (NUM1)
	N_{eq}	NEQ	The number of coupled governing equations actually being solved. (NUM1)
	N_v	NV	The number of grid points in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, NV = N_2 for the first sweep and N_1 for the second and third sweeps. (NUM1)
*	N_1	N1	The number of grid points in the ξ direction. (NUM1)
	N_1	NPT1	The number of grid points in the ξ direction used in computing coefficients (only for non-periodic boundary conditions.) (NUM1)
	$N_1 + 1$	NPT1	The number of grid points in the ξ direction used in computing coefficients (only for spatially periodic boundary conditions.) (NUM1)
*	N_2	N2	The number of grid points in the η direction. (NUM1)
	N_2	NPT2	The number of grid points in the η direction used in computing coefficients (only for non-periodic boundary conditions.) (NUM1)
	$N_2 + 1$	NPT2	The number of grid points in the η direction used in computing coefficients (only for spatially periodic boundary conditions.) (NUM1)
*	N_3	N3	The number of grid points in the ζ direction. (NUM1)
	N_3	NPT3	The number of grid points in the ζ direction used in computing coefficients (only for non-periodic boundary conditions.) (NUM1)

	$N_3 + 1$	NPT3	The number of grid points in the ζ direction used in computing coefficients (only for spatially periodic boundary conditions.) (NUM1)
	p	P(I1,I2,I3)	Static pressure at time level n. (FLOW1)
	p_r	PR	Dimensional reference static pressure, $\rho_r \overline{R} T_r / g_c$. (FLOW1)
*	p_0	P0	Initial static pressure. (IC1)
	$\partial p / \partial E_T$	DPDET(I)	The derivative of p with respect to E_T, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial p / \partial \rho$	DPDRHO(I)	The derivative of p with respect to ρ, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial p / \partial (\rho u)$	DPDRU(I)	The derivative of p with respect to ρu, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial p / \partial (\rho v)$	DPDRV(I)	The derivative of p with respect to ρv, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial p / \partial (\rho w)$	DPDRW(I)	The derivative of p with respect to ρw, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	P_k	VORT(I1,I2,I3)	Production rate of turbulent kinetic energy. (TURB1)
*	Pr_{l_r}	PRLR	Reference laminar Prandtl number, $c_{p_r} \mu_r / k_r$, where $c_{p_r} = \gamma_r \overline{R} / (\gamma_r - 1)$. (FLOW1)
	Pr_r	PRR	Reference Prandtl number, $\mu_r u_r^2 / k_r T_r$. (FLOW1)
*	Pr_t	PRT	Turbulent Prandtl number, or, if non-positive, a flag indicating the use of a variable turbulent Prandtl number. (TURB1)
	ΔQ_{avg}	CHGAVG(I)	Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps.[5] The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (TIME1)
	ΔQ_{max}	CHGMAX(I,J)	Maximum change in absolute value of the dependent variables over a single time step.[5] The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables, and J = 1 to NITAVG, the number of time steps used in the moving average option for determining convergence. (TIME1)
*	r_{max}	RMAX	Maximum r coordinate coordinate for cylindrical grid option. (GMTRY1)

[5] For the energy equation, the change in E_T is divided by $E_{T_r} = \rho_r \overline{R} T_r / (\gamma_r - 1) + u_r^2/2$, so that it is the same order of magnitude as the other conservation variables.

*	r_{min}	RMIN	Minimum r coordinate coordinate for cylindrical grid option. (GMTRY1)
	R_{avg}	RESAVG(J,K)	The average absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
	R_{L_2}	RESL2(J,K)	The L_2 norm of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
	R_{max}	RESMAX(J,K)	The maximum absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)
*	$\overline{R}$	RG	Dimensional gas constant. (FLOW1)
	R	RGAS	Non-dimensional gas constant. (FLOW1)
*	Re_r	RER	Reference Reynolds number, $\rho_r u_r L_r / \mu_r$. (FLOW1)
	S	SVECT(IV,IS,J)	Subvector of source terms in the block tridiagonal system of equations. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	Δt	DT(I)	The time step size, when specified directly as input. I is the time step sequence number, and runs from 1 to NTSEQ. (TIME1)
	T	T(I1,I2,I3)	Static temperature at time level n. (FLOW1)
	T	TL(I1,I2,I3)	Static temperature at previous or intermediate time level. (FLOW1)
	$\partial T/\partial E_T$	DTDET(I)	The derivative of T with respect to E_T, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial T/\partial \rho$	DTDRHO(I)	The derivative of T with respect to ρ, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial T/\partial(\rho u)$	DTDRU(I)	The derivative of T with respect to ρu, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
	$\partial T/\partial(\rho v)$	DTDRV(I)	The derivative of T with respect to ρv, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

	$\partial T/\partial(\rho w)$	DTDRW(I)	The derivative of T with respect to ρw, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)
*	T_r	TR	Dimensional reference temperature. (FLOW1)
*	T_0	T0	Initial static temperature. (IC1)
	u	U(I1,I2,I3)	Velocity in the Cartesian x direction at time level n. (FLOW1)
	u	UL(I1,I2,I3)	Velocity in the Cartesian x direction at previous or intermediate time level. (FLOW1)
*	u_r	UR	Dimensional reference velocity. (FLOW1)
*	u_0	U0	Initial velocity in the Cartesian x direction. (IC1)
	v	V(I1,I2,I3)	Velocity in the Cartesian y direction at time level n. (FLOW1)
	v	VL(I1,I2,I3)	Velocity in the Cartesian y direction at previous or intermediate time level. (FLOW1)
*	v_0	V0	Initial velocity in the Cartesian y direction. (IC1)
	w	W(I1,I2,I3)	Velocity in the Cartesian z direction at time level n. (FLOW1)
	w	WL(I1,I2,I3)	Velocity in the Cartesian z direction at previous or intermediate time level. (FLOW1)
*	w_0	W0	Initial velocity in the Cartesian z direction. (IC1)
	$\Delta\hat{\mathbf{W}}^n$ or $\Delta\hat{\mathbf{W}}^*$	DW(I1,I2,I3,I)	Unknown vector in the LU solution of the k-ε equations. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively. (NUM2)
	x	X(I1,I2,I3)	Cartesian x coordinate. (METRIC1)
*	x_{max}	XMAX	Maximum x coordinate for Cartesian or cylindrical grid option. (GMTRY1)
*	x_{min}	XMIN	Minimum x coordinate for Cartesian or cylindrical grid option. (GMTRY1)
	y	Y(I1,I2,I3)	Cartesian y coordinate. (METRIC1)
*	y_{max}	YMAX	Maximum y coordinate for Cartesian grid option. (GMTRY1)
*	y_{min}	YMIN	Minimum y coordinate for Cartesian grid option. (GMTRY1)
	y^+	YPLUSD(I1,I2,I3)	Non-dimensional distance from the nearest solid wall. (TURB20)
	z	Z(I1,I2,I3)	Cartesian z coordinate. (METRIC1)

*	z_{max}	ZMAX	Maximum z coordinate for Cartesian grid option. (GMTRY1)
*	z_{min}	ZMIN	Minimum z coordinate for Cartesian grid option. (GMTRY1)
*	ε	EPS(I)	Convergence level to be reached. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (TIME1)
	ε	E(I1,I2,I3)	Turbulent dissipation rate at time level n. (TURB20)
	ε	EL(I1,I2,I3)	Turbulent dissipation rate at previous or intermediate time level. (TURB20)
*	$\varepsilon_E^{(2)}$	CAVS2E(I)	Second order explicit artificial viscosity coefficient in constant coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	$\varepsilon_E^{(4)}$	CAVS4E(I)	Fourth order explicit artificial viscosity coefficient in constant coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	ε_I	CAVS2I(I)	Second order implicit artificial viscosity coefficient in constant coefficient model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
	ζ_t	ZETAT(I1,I2,I3)	The derivative of the computational coordinate ζ with respect to untransformed time t. (METRIC1)
	ζ_t	METT(IV,IS)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate t (third ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	ζ_x	ZETAX(I1,I2,I3)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate x. (METRIC1)
	ζ_x	METX(IV,IS)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate x (third ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	ζ_y	ZETAY(I1,I2,I3)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate y. (METRIC1)
	ζ_y	METY(IV,IS)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate y (third ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)

ζ_z	ZETAZ(I1,I2,I3)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate z. (METRIC1)
ζ_z	METZ(IV,IS)	The derivative of the computational coordinate ζ with respect to the Cartesian coordinate z (third ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
$\Delta\zeta$	DEL	Computational grid spacing in the ζ direction (third ADI sweep only.) (NUM1)
$\Delta\zeta$	DZETA	Computational grid spacing in the ζ direction. (NUM1)
η_t	ETAT(I1,I2,I3)	The derivative of the computational coordinate η with respect to untransformed time t. (METRIC1)
η_t	METT(IV,IS)	The derivative of the computational coordinate η with respect to untransformed time t (second ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
η_x	ETAX(I1,I2,I3)	The derivative of the computational coordinate η with respect to the Cartesian coordinate x. (METRIC1)
η_x	METX(IV,IS)	The derivative of the computational coordinate η with respect to the Cartesian coordinate x (second ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
η_y	ETAY(I1,I2,I3)	The derivative of the computational coordinate η with respect to the Cartesian coordinate y. (METRIC1)
η_y	METY(IV,IS)	The derivative of the computational coordinate η with respect to the Cartesian coordinate y (second ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
η_z	ETAZ(I1,I2,I3)	The derivative of the computational coordinate η with respect to the Cartesian coordinate z. (METRIC1)
η_z	METZ(IV,IS)	The derivative of the computational coordinate η with respect to the Cartesian coordinate z (second ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)

	$\Delta\eta$	DEL	Computational grid spacing in the η direction (second ADI sweep only.) (NUM1)
	$\Delta\eta$	DETA	Computational grid spacing in the η direction. (NUM1)
*	κ	CVK	Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein turbulence models. (TURB1)
*	κ_2	CAVS2E(I)	User-specified constant in nonlinear coefficient artificial viscosity model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	κ_4	CAVS4E(I)	User-specified constant in nonlinear coefficient artificial viscosity model. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations. (NUM1)
*	γ_r	GAMR	Reference ratio of specific heats, c_{p_r}/c_{v_r}. (FLOW1)
	λ	LA(I1,I2,I3)	Effective second coefficient of viscosity at time level n (usually assumed equal to $-2\mu/3$.) This is a real variable. (FLOW1)
	μ	MU(I1,I2,I3)	Effective viscosity coefficient at time level n. This is a real variable. (FLOW1)
*	μ_r	MUR	Dimensional reference viscosity coefficient. This is a real variable. (FLOW1)
	μ_T	MUT(I1,I2,I3)	Turbulent viscosity coefficient at time level n. This is a real variable. (FLOW1)
	μ_t	MUTL(I1,I2,I3)	Turbulent viscosity coefficient at previous or intermediate time level. This is a real variable. (TURB20)
	ξ_t	XIT(I1,I2,I3)	The derivative of the computational coordinate ξ with respect to untransformed time t. (METRIC1)
	ξ_t	METT(IV,IS)	The derivative of the computational coordinate ξ with respect to untransformed time t (first ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	ξ_x	XIX(I1,I2,I3)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate x. (METRIC1)
	ξ_x	METX(IV,IS)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate x (first ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	ξ_y	XIY(I1,I2,I3)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate y. (METRIC1)
	ξ_y	METY(IV,IS)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate y (first ADI sweep only.) IS is the

			grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	ξ_z	XIZ(I1,I2,I3)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate z. (METRIC1)
	ξ_z	METZ(IV,IS)	The derivative of the computational coordinate ξ with respect to the Cartesian coordinate z (first ADI sweep only.) IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to $N_v - 1$. This is a real variable. (METRIC1)
	$\Delta\xi$	DEL	Computational grid spacing in the ξ direction (first ADI sweep only.) (NUM1)
	$\Delta\xi$	DXI	Computational grid spacing in the ξ direction. (NUM1)
	ρ	RHO(I1,I2,I3)	Static density at time level n. (FLOW1)
	ρ	RHOL(I1,I2,I3)	Static density at previous or intermediate time level. (FLOW1)
*	ρ_r	RHOR	Dimensional reference density. (FLOW1)
*	σ_k	SIGK	Constant used in the diffusion term of the k equation. (TURB20)
*	σ_ε	SIGE	Constant used in the diffusion term of the ε equation. (TURB20)
	τ	TAU(I1,I2,I3)	Current value of the time marching parameter. (TIME1)
	$\Delta\tau$	DTAU(I1,I2,I3)	Computational time step size. (TIME1)
*	θ_{max}	THMAX	Maximum θ coordinate in degrees for cylindrical grid option. (GMTRY1)
*	θ_{min}	THMIN	Minimum θ coordinate in degrees for cylindrical grid option. (GMTRY1)
*	$\theta_1,\ \theta_2$	THC(I)	A two-element array specifying the time difference centering parameters used for the continuity equation. (NUM1)
*	$\theta_1,\ \theta_2$	THKE(I)	A two-element array specifying the time difference centering parameters used for the k-ε equations. (NUM2)
*	$\theta_1,\ \theta_2,\ \theta_3$	THE(I)	A three-element array specifying the time difference centering parameters used for the energy equation. (NUM1)
*	$\theta_1,\ \theta_2,\ \theta_3$	THX(I)	A three-element array specifying the time difference centering parameters used for the x-momentum equation. (NUM1)
*	$\theta_1,\ \theta_2,\ \theta_3$	THY(I)	A three-element array specifying the time difference centering parameters used for the y-momentum equation. (NUM1)

* $\theta_1,\ \theta_2,\ \theta_3$	THZ(I)	A three-element array specifying the time difference centering parameters used for the z-momentum equation. (NUM1)
$\lvert\bar{\Omega}\rvert$	VORT(I1,I2,I3)	Total vorticity magnitude. (TURB1)

4.0 PROTEUS SUBPROGRAMS

In this section, each subprogram in *Proteus* is described, first in summary, then in detail. The summary is simply a list of the subprograms with a brief description of the purpose of each one. The detailed description includes, for each subprogram, a list of the subprograms that reference it, and a list of the subprograms that it references. The Fortran variables that are input to and output from each subprogram are defined. And finally, details of the computations being done within each subprogram are presented.

4.1 SUBPROGRAM SUMMARY

The following table presents a brief description of the purpose of each subprogram in the *Proteus* code.

Proteus Subprogram Summary

Subprogram	Purpose
ADI	Manage the block tridiagonal inversion.
AVISC1	Compute constant coefficient artificial viscosity.
AVISC2	Compute nonlinear coefficient artificial viscosity.
BCDENS	Compute density boundary conditions.
BCELIM	Eliminate off-diagonal coefficient submatrices resulting from three-point boundary conditions.
BCF	Compute user-written boundary conditions.
BCFLIN	User-supplied routine for linearization of user-supplied boundary conditions.
BCGEN	Manage computation of boundary conditions.
BCGRAD	Compute gradients with respect to ξ, η, and ζ.
BCIMET	Compute inverse metrics at a point in the current sweep direction.
BCMET	Compute various metric functions for normal gradient boundary conditions.
BCNVEL	Compute normal direction velocity boundary conditions.
BCPRES	Compute pressure boundary conditions.
BCQ	Compute conservation variable boundary conditions.
BCSET	Set various boundary condition parameters and flags.
BCTEMP	Compute temperature boundary conditions.
BCUVEL	Compute x-velocity boundary conditions.
BCVN	Compute velocity normal to a surface.
BCVVEL	Compute y-velocity boundary conditions.
BCV1	Compute ξ-velocity.
BCV2	Compute η-velocity.
BCV3	Compute ζ-velocity.
BCWVEL	Compute z-velocity boundary conditions.
BC1VEL	Compute ξ-velocity boundary conditions.
BC2VEL	Compute η-velocity boundary conditions.

Proteus Subprogram Summary	
Subprogram	**Purpose**
BC3VEL	Compute ζ-velocity boundary conditions.
BLIN	Compute inner layer turbulent viscosity, using the Baldwin-Lomax model.
BLKOUT	Print coefficient blocks at specified indices in the ξ, η, and ζ directions.
BLK4	Solve 4×4 block tridiagonal system of equations.
BLK4P	Solve 4×4 periodic block tridiagonal system of equations.
BLK5	Solve 5×5 block tridiagonal system of equations.
BLK5P	Solve 5×5 periodic block tridiagonal system of equations.
BLOCK DATA	Set default values for input parameters, plus a few other parameters.
BLOUT	Compute outer layer turbulent viscosity, using the Baldwin-Lomax model.
BVUP	Update first and second sweep boundary values after third sweep.
COEFC	Compute coefficients and source terms for the continuity equation.
COEFE1	Compute coefficients and source terms for the energy equation.
COEFE2	Compute source terms for the energy equation.
COEFX	Compute coefficients and source terms for the x-momentum equation.
COEFY	Compute coefficients and source terms for the y-momentum equation.
COEFZ	Compute coefficients and source terms for the z-momentum equation.
CONV	Test computed flow field for convergence.
CUBIC	Interpolation using Ferguson's parametric cubic.
EQSTAT	Use equation of state to compute pressure, temperature, and their derivatives with respect to the dependent variables.
EXEC	Manage solution of governing equations.
EXECT	Manage solution of the k-ε equations.
FILTER	Rearrange rows of the boundary condition coefficient submatrices and the source term subvector to eliminate any zeroes on the diagonal.
FTEMP	Compute auxiliary variables that are functions of temperature.
GATHER	Create a vector containing specified elements of an input vector. This is a Cray Linear Algebra routine.
GEOM	Manage computation of grid and metric parameters.
INIT	Get user-defined initial flow field.
INITC	Set up consistent initial conditions based on data from INIT.
INPUT	Read and print input, perform various initializations.
ISAMAX	Find the first index corresponding to the largest absolute value of the elements of an vector. This is a Cray search routine.
ISAMIN	Find the first index corresponding to the smallest absolute value of the elements of an vector. This is a Cray search routine.
ISRCHEQ	Find the first index in an array whose element is equal to a specified value. This is a Cray search routine.

Proteus Subprogram Summary	
Subprogram	Purpose
ISRCHFGT	Find the first index in an array whose element is greater than a specified value. This is a Cray search routine.
ISRCHFLT	Find the first index in an array whose element is less than a specified value. This is a Cray search routine.
KEINIT	Get user-defined initial conditions for k and ε.
MAIN	Manage overall solution.
METS	Compute metrics of nonorthogonal grid transformation.
OUTPUT	Manage printing of output.
OUTW	Compute and print parameters at boundaries.
PAK	Manage packing and/or interpolation of grid points.
PERIOD	Define extra line of data for use in computing coefficients for spatially periodic boundary conditions.
PLOT	Write files for post-processing by CONTOUR or PLOT3D plotting programs.
PRODCT	Compute production term for the k-ε turbulence model.
PRTHST	Print convergence history.
PRTOUT	Print output.
RESID	Compute residuals and write convergence history file.
REST	Read and/or write restart file.
ROBTS	Pack points along a line using Roberts transformation.
SASUM	Compute the sum of the absolute values of the elements of a vector. This is a Cray BLAS routine.
SGEFA	Factor a matrix using Gaussian elimination. This is a Cray LINPACK routine.
SGESL	Solve the matrix equation $\mathbf{Ax} = \mathbf{B}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{B}$ using the factors computed by SGEFA. This is a Cray LINPACK routine.
SNRM2	Compute the L_2 norm of a vector. This is a Cray BLAS routine.
SWDOWN	Compute coefficients and source terms, and solve the k-ε equations for the downward LU sweep.
SWUP	Compute coefficients and source terms, and solve the k-ε equations for the upward LU sweep.
TBC	Set time-dependent boundary condition values.
TIMSTP	Set computational time step.
TREMAIN	Get CPU time remaining for the job. This is a Cray Fortran routine.
TURBBL	Manage computation of turbulence parameters using Baldwin-Lomax algebraic model.
TURBCH	Manage computation of turbulence parameters using the Chien k-ε model.
UPDATE	Update flow variables after each ADI sweep.
UPDTKE	Update k and ε after each time step.
VORTEX	Compute magnitude of total vorticity.
WHENFLT	Find all indices in an array whose elements are less than a specified value. This is a Cray search routine.

Proteus Subprogram Summary	
Subprogram	Purpose
YPLUSN	Compute the distance to the nearest solid wall.

4.2 SUBPROGRAM DETAILS

The subprograms used in *Proteus* are described in detail in the remainder of this section. A few additional words are necessary about the input and output descriptions. The description of the input to each subprogram includes all Fortran variables actually used by the subprogram that are defined outside the subprogram. Variables defined and used inside the subprogram are not listed as input. In addition, common block variables that are merely passed through to lower level routines are not listed. Variables marked with an asterisk are user-specified namelist input variables.

Similarly, the output description includes only those variables computed inside the subprogram and used outside the subprogram. It does not include common block variables computed by lower level routines. In general, variables defined inside the subprogram that are used by lower level routines are listed as output, even if they are not needed after control is returned to the calling program.

Variables entering or leaving a subprogram through an argument list are defined in detail. However, most of the Fortran variables listed in the input and output descriptions are contained in common blocks, and are defined in detail in Section 3.0. For that reason, they are defined only briefly in this section.

Subroutine ADI		
Called by	Calls	Purpose
EXEC	BLKOUT BLK4 BLK4P BLK5 BLK5P	Manage the block tridiagonal inversion.

Input

*	IDEBUG	Debug flags.
*	IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
	ISWEEP	Current ADI sweep number.
	IT	Current time step number n.
	KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
	NEQ	Number of coupled equations being solved, N_{eq}.
*	NOUT	Unit number for standard output.
	NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.

Output

None.

Description

For each ADI sweep, subroutine ADI calls the appropriate block solver. The choice is determined by the number of equations being solved, and by the presence or absence of spatially periodic boundary conditions in the sweep direction.

Remarks

1. This subroutine generates the output for the IDEBUG(1), IDEBUG(5), and IDEBUG(6) options.

Subroutine AVISC1		
Called by	Calls	Purpose
EXEC	BLKOUT	Compute constant coefficient artificial viscosity.

Input

	A, B, C	Coefficient submatrices **A**, **B**, and **C** without artificial viscosity.
*	CAVS2E, CAVS4E, CAVS2I	Artificial viscosity coefficients $\varepsilon_E^{(2)}$, $\varepsilon_E^{(4)}$, and ε_I.
	DTAU	Time step $\Delta\tau$.
*	IAV2E, IAV4E, IAV2I	Flags for second-order explicit, fourth-order explicit, and second-order implicit artificial viscosity.
*	IDEBUG	Debug flags.
*	IHSTAG	Flag for constant stagnation enthalpy option.
*	IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
	ISWEEP	Current ADI sweep number.
	IT	Current time step number n.
	I2, I3	Grid indices j and k, in the η and ζ directions.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations.
*	NOUT	Unit number for standard output.
	NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
	NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
	RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
	S	Source term subvector **S** without artificial viscosity.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** with artificial viscosity.
S	Source term subvector **S** with artificial viscosity.

Description

Subroutine AVISC1 adds explicit and/or implicit artificial viscosity to the governing equations, using the constant coefficient model of Steger (1978), as presented by Pulliam (1986b). The model is described in Section 8.1 of Volume 1. The explicit artificial viscosity may be second and/or fourth order, and is added only during the first ADI sweep. The implicit artificial viscosity is second order, and is added during all three sweeps.

The fourth-order explicit artificial viscosity is implemented in Fortran by redefining the source term subvector as

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} - \frac{\varepsilon_E^{(4)} \Delta\tau_{i,j,k}}{J_{i,j,k}} [(\nabla_\xi \Delta_\xi)^2 \mathbf{Q}_{i,j,k} + (\nabla_\eta \Delta_\eta)^2 \mathbf{Q}_{i,j,k} + (\nabla_\zeta \Delta_\zeta)^2 \mathbf{Q}_{i,j,k}]$$

where i, j, and k vary from 3 to NPT1 − 2, NPT2 − 2, and NPT3 − 2, respectively. At grid points adjacent to boundaries the fourth-order differences in the above equation cannot be used, and therefore are replaced by second-order differences. Thus, at $i = 2$ and at $i = \text{NPT1} - 1$, with j and k varying from 3 to NPT2 − 2 and NPT3 − 2,

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} + \frac{\varepsilon_E^{(4)} \Delta\tau_{i,j,k}}{J_{i,j,k}} [\nabla_\xi \Delta_\xi \mathbf{Q}_{i,j,k} - (\nabla_\eta \Delta_\eta)^2 \mathbf{Q}_{i,j,k} - (\nabla_\zeta \Delta_\zeta)^2 \mathbf{Q}_{i,j,k}]$$

Similarly, at $j = 2$ and at $j = \text{NPT2} - 1$, with i and k varying from 3 to NPT1 − 1 and NPT3 − 2,

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} + \frac{\varepsilon_E^{(4)} \Delta\tau_{i,j,k}}{J_{i,j,k}} [-(\nabla_\xi \Delta_\xi)^2 \mathbf{Q}_{i,j,k} + \nabla_\eta \Delta_\eta \mathbf{Q}_{i,j,k} - (\nabla_\zeta \Delta_\zeta)^2 \mathbf{Q}_{i,j,k}]$$

And, at $k = 2$ and at $k = \text{NPT3} - 1$, with i and j varying from 3 to NPT1 − 1 and NPT2 − 2,

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} + \frac{\varepsilon_E^{(4)} \Delta\tau_{i,j,k}}{J_{i,j,k}} [-(\nabla_\xi \Delta_\xi)^2 \mathbf{Q}_{i,j,k} - (\nabla_\eta \Delta_\eta)^2 \mathbf{Q}_{i,j,k} + (\nabla_\zeta \Delta_\zeta) \mathbf{Q}_{i,j,k}]$$

The second-order explicit artificial viscosity is implemented in Fortran by redefining the source term subvector as

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} + \frac{\varepsilon_E^{(2)} \Delta\tau_{i,j,k}}{J_{i,j,k}} (\nabla_\xi \Delta_\xi \mathbf{Q}_{i,j,k} + \nabla_\eta \Delta_\eta \mathbf{Q}_{i,j,k} + \nabla_\zeta \Delta_\zeta \mathbf{Q}_{i,j,k})$$

where i, j, and k vary from 2 to NPT1 − 1, NPT2 − 1, and NPT3 − 1, respectively.

The second-order implicit artificial viscosity for the first ADI sweep is implemented in Fortran by redefining the cc·fficient block submatrices as

$$\mathbf{A}_{i,j,k} = \mathbf{A}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i-1,j,k}$$

$$\mathbf{B}_{i,j,k} = \mathbf{B}_{i,j,k} + 2 \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j,k}$$

$$\mathbf{C}_{i,j,k} = \mathbf{C}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i+1,j,k}$$

where i, j, and k vary from 2 to NPT1 − 1, NPT2 − 1, and NPT3 − 1, respectively. Similarly, for the second sweep,

$$\mathbf{A}_{i,j,k} = \mathbf{A}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j-1,k}$$

$$\mathbf{B}_{i,j,k} = \mathbf{B}_{i,j,k} + 2 \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j,k}$$

$$\mathbf{C}_{i,j,k} = \mathbf{C}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j+1,k}$$

And, for the third sweep,

$$\mathbf{A}_{i,j,k} = \mathbf{A}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j,k-1}$$

$$\mathbf{B}_{i,j,k} = \mathbf{B}_{i,j,k} + 2\frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j,k}$$

$$\mathbf{C}_{i,j,k} = \mathbf{C}_{i,j,k} - \frac{\varepsilon_I \Delta\tau_{i,j,k}}{J_{i,j,k}} J_{i,j,k+1}$$

Remarks

1. The sign in front of each artificial viscosity term depends on the sign of the "*i,j,k*" term in the difference formula. See Section 8.1 of Volume 1 for details.
2. The coding to add artificial viscosity to the energy equation is separate from the coding for the remaining equations, and is bypassed if it is not being solved.
3. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep (which includes all the explicit artificial viscosity) the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3).
4. For spatially periodic boundary conditions in the ξ direction, fourth-order differences could be used at $i = 2$ and at $i = \mathrm{NPT1} - 1$ $(= N_1)$. A similar situation occurs with spatially periodic boundary conditions in the η and ζ directions. The logic to do this has not been coded, however, and at these points second-order differences are still used, as described above.
5. This subroutine generates the output for the IDEBUG(2) option.

Subroutine AVISC2		
Called by	Calls	Purpose
EXEC	BLKOUT	Compute nonlinear coefficient artificial viscosity.

Input

* CAVS2E, CAVS4E	User-specified coefficients κ_2 and κ_4.
CP, CV	Specific heats c_p and c_v at time level n.
DTAU	Time step $\Delta\tau$.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
* IAV2E, IAV4E	Flags for second-order and fourth-order explicit artificial viscosity.
* IDEBUG	Debug flags.
* IHSTAG	Flag for constant stagnation enthalpy option.
* IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
ISWEEP	Current ADI sweep number.
IT	Current time step number n.
I2, I3	Grid indices j and k, in the η and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations.
* NOUT	Unit number for standard output.
NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
P, T	Static pressure p and temperature T at time level n.
RGAS	Gas constant R.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
S	Source term subvector **S** without artificial viscosity.
XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

S	Source term subvector **S** with artificial viscosity.

Description

Subroutine AVISC2 adds explicit artificial viscosity to the governing equations, using the nonlinear coefficient model of Jameson, Schmidt, and Turkel (1981), as presented by Pulliam (1986b). The model is described in Section 8.2 of Volume 1. Implicit artificial viscosity is not normally used in combination with

the explicit nonlinear coefficient model. The explicit artificial viscosity is added only during the first ADI sweep.

The artificial viscosity in the ξ direction is computed first, at the η-indices $j = 2$ to NPT2 − 1 and ζ-indices $k = 2$ to NPT3 − 1. The spectral radius term $\psi_{i,j,k}$ and the pressure gradient scaling factor $\sigma_{i,j,k}$ are computed and stored in local one-dimensional arrays for $i = 1$ to NPT1. Special formulas are used to compute σ near boundaries, as described in Section 8.2 of Volume 1.

The second-order artificial viscosity is added first, and is implemented in Fortran by redefining the source term subvector as

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} + \nabla_\xi \left\{ \left[\left(\frac{\psi}{J} \right)_{i+1,j,k} + \left(\frac{\psi}{J} \right)_{i,j,k} \right] (\varepsilon_\xi^{(2)})_{i,j,k} \Delta_\xi \mathbf{Q}_{i,j,k} \right\}$$

Or, after evaluating the differences,

$$\begin{aligned} \mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} &+ \left[\left(\frac{\psi}{J} \right)_{i+1,j,k} + \left(\frac{\psi}{J} \right)_{i,j,k} \right] (\varepsilon_\xi^{(2)})_{i,j,k} (\mathbf{Q}_{i+1,j,k} - \mathbf{Q}_{i,j,k}) \\ &- \left[\left(\frac{\psi}{J} \right)_{i,j,k} + \left(\frac{\psi}{J} \right)_{i-1,j,k} \right] (\varepsilon_\xi^{(2)})_{i-1,j,k} (\mathbf{Q}_{i,j,k} - \mathbf{Q}_{i-1,j,k}) \end{aligned}$$

where i varies from 2 to NPT1 − 1.

The fourth-order explicit artificial viscosity is added next, and is implemented similarly by redefining the source term subvector as

$$\mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} - \nabla_\xi \left\{ \left[\left(\frac{\psi}{J} \right)_{i+1,j,k} + \left(\frac{\psi}{J} \right)_{i,j,k} \right] (\varepsilon_\xi^{(4)})_{i,j,k} \Delta_\xi \nabla_\xi \Delta_\xi \mathbf{Q}_{i,j,k} \right\}$$

Or, after evaluating the differences,

$$\begin{aligned} \mathbf{S}_{i,j,k} = \mathbf{S}_{i,j,k} &- \left[\left(\frac{\psi}{J} \right)_{i+1,j,k} + \left(\frac{\psi}{J} \right)_{i,j,k} \right] (\varepsilon_\xi^{(4)})_{i,j,k} (\mathbf{Q}_{i+2,j,k} - 3\mathbf{Q}_{i+1,j,k} + 3\mathbf{Q}_{i,j,k} - \mathbf{Q}_{i-1,j,k}) \\ &+ \left[\left(\frac{\psi}{J} \right)_{i,j,k} + \left(\frac{\psi}{J} \right)_{i-1,j,k} \right] (\varepsilon_\xi^{(4)})_{i-1,j,k} (\mathbf{Q}_{i+1,j,k} - 3\mathbf{Q}_{i,j,k} + 3\mathbf{Q}_{i-1,j,k} - \mathbf{Q}_{i-2,j,k}) \end{aligned}$$

where i varies from 3 to NPT1 − 2. Special formulas are used at $i = 2$ and at $i =$ NPT1 − 1, as described in Section 8.2 of Volume 1.

The explicit artificial viscosity in the η and ζ directions is then implemented in a manner analogous to that just described for the explicit artificial viscosity in the ξ direction.

<u>Remarks</u>

1. The sign in front of each artificial viscosity term depends on the sign of the "i,j,k" term in the difference formula. See Section 8.1 of Volume 1 for details.
2. The coding to add artificial viscosity to the energy equation is separate from the coding for the remaining equations, and is bypassed if it is not being solved.
3. The subscripts on the Fortran variable S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction.

For the first sweep (which includes all the explicit artificial viscosity) the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3).

4. For spatially periodic boundary conditions, the need for special formulas near boundaries could be eliminated. The logic to do this has not been coded, however.
5. This subroutine generates the output for the IDEBUG(2) option.

Subroutine BCDENS (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute density boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR	Array index associated with the dependent variable ρ.
RHO	Static density ρ at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCDENS computes coefficients and source terms for density boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular density boundary conditions in *Proteus*.[6]

[6] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

No Change From Initial Conditions, $\Delta\rho = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial\hat{\mathbf{Q}} = J\partial g/\partial\mathbf{Q}$, we get simply

$$J_{i,j,k}\Delta\hat{\rho}^n_{i,j,k} = 0$$

Specified Static Density, $\rho = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\Delta\hat{\rho}^n_{i,j,k} = f^{n+1}_{i,j,k} - \rho^n_{i,j,k}$$

Specified Two-Point Density Gradient in Coordinate Direction, $\partial\rho/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j,k}\Delta\hat{\rho}^n_{1,j,k} + J_{2,j,k}\Delta\hat{\rho}^n_{2,j,k} = (\Delta\xi)f^{n+1}_{1,j,k} + \rho^n_{1,j,k} - \rho^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j,k}\Delta\hat{\rho}^n_{N_1-1,j,k} + J_{N_1,j,k}\Delta\hat{\rho}^n_{N_1,j,k} = (\Delta\xi)f^{n+1}_{N_1,j,k} + \rho^n_{N_1-1,j,k} - \rho^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Density Gradient in Coordinate Direction, $\partial\rho/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$\begin{aligned}-3J_{1,j,k}\Delta\hat{\rho}^n_{1,j,k} + 4J_{2,j,k}\Delta\hat{\rho}^n_{2,j,k} - J_{3,j,k}\Delta\hat{\rho}^n_{3,j,k} &= \\ 2(\Delta\xi)f^{n+1}_{1,j,k} + 3\rho^n_{1,j,k} - 4\rho^n_{2,j,k} + \rho^n_{3,j,k}\end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned}J_{N_1-2,j,k}\Delta\hat{\rho}^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\Delta\hat{\rho}^n_{N_1-1,j,k} + 3J_{N_1,j,k}\Delta\hat{\rho}^n_{N_1,j,k} &= \\ 2(\Delta\xi)f^{n+1}_{N_1,j,k} - \rho^n_{N_1-2,j,k} + 4\rho^n_{N_1-1,j,k} - 3\rho^n_{N_1,j,k}\end{aligned}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Density Gradient in Normal Direction, $\nabla\rho \bullet \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\begin{aligned}&-J_{1,j,k}\Delta\hat{\rho}^n_{1,j,k} + J_{2,j,k}\Delta\hat{\rho}^n_{2,j,k} = \\ &\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta\rho^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta\rho^n_{1,j,k}\right] \\ &+\rho^n_{1,j,k} - \rho^n_{2,j,k}\end{aligned}$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\, \Delta\hat{\rho}^n_{N_1-1,j,k} + J_{N_1,j,k}\, \Delta\hat{\rho}^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}} \left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}} \delta_\eta \rho^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}} \delta_\zeta \rho^n_{N_1,j,k} \right] \\
&+ \rho^n_{N_1-1,j,k} - \rho^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Density Gradient in Normal Direction, $\nabla\rho \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\, \Delta\hat{\rho}^n_{1,j,k} + 4J_{2,j,k}\, \Delta\hat{\rho}^n_{2,j,k} - J_{3,j,k}\, \Delta\hat{\rho}^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}} \left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}} \delta_\eta \rho^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}} \delta_\zeta \rho^n_{1,j,k} \right] \\
&+ 3\rho^n_{1,j,k} - 4\rho^n_{2,j,k} + \rho^n_{3,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\, \Delta\hat{\rho}^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\, \Delta\hat{\rho}^n_{N_1-1,j,k} + 3J_{N_1,j,k}\, \Delta\hat{\rho}^n_{N_1,j,k} = \\
&\frac{2\Delta\xi}{m_{N_1,j,k}} \left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}} \delta_\eta \rho^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}} \delta_\zeta \rho^n_{N_1,j,k} \right] \\
&- \rho^n_{N_1-2,j,k} + 4\rho^n_{N_1-1,j,k} - 3\rho^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of Static Density

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$
J_{1,j,k}\, \Delta\hat{\rho}^n_{1,j,k} - 2J_{2,j,k}\, \Delta\hat{\rho}^n_{2,j,k} + J_{3,j,k}\, \Delta\hat{\rho}^n_{3,j,k} = - \rho^n_{1,j,k} + 2\rho^n_{2,j,k} - \rho^n_{3,j,k}
$$

At the $\xi = 1$ boundary,

$$
J_{N_1-2,j,k}\, \Delta\hat{\rho}^n_{N_1-2,j,k} - 2J_{N_1-1,j,k}\, \Delta\hat{\rho}^n_{N_1-1,j,k} + J_{N_1,j,k}\, \Delta\hat{\rho}^n_{N_1,j,k} = - \rho^n_{N_1-2,j,k} + 2\rho^n_{N_1-1,j,k} - \rho^n_{N_1,j,k}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent density boundary condition is specified.

Subroutine BCELIM		
Called by	Calls	Purpose
EXEC	SGEFA SGESL	Eliminate off-diagonal coefficient submatrices resulting from three-point boundary conditions.

Input

A, B, C	Coefficient submatrices **A**, **B**, and **C** before eliminating off-diagonal blocks.
IBCELM	Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the ξ and/or η directions; 0 if elimination is not necessary, 1 if it is.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
NEQ	Number of coupled equations being solved, N_{eq}.
NEQP	Dimensioning parameter specifying maximum number of coupled equations allowed.
NPTS	Number of grid points in the sweep direction, N.
S	Source term subvector **S** before eliminating off-diagonal blocks.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** after eliminating off-diagonal blocks.
S	Source term subvector **S** after eliminating off-diagonal blocks.

Description

Subroutine BCELIM eliminates the off-diagonal coefficient submatrices that result from the application of three-point boundary conditions. This is necessary when three-point gradients are specified in the coordinate or normal direction, and when linear extrapolation is used. The procedure is described in Section 7.2.1 of Volume 1.

Remarks

1. Subroutines SGEFA and SGESL are Cray LINPACK routines. In general terms, if the Fortran arrays A and B represent **A** and **B**, where **A** is a square N by N matrix and **B** is a matrix (or vector) with NCOL columns, and if the leading dimension of the Fortran array A is LDA, then the Fortran sequence

```
      call sgefa (a,lda,n,ipvt,info)
      do 10 j = 1,ncol
      call sgesl (a,lda,n,ipvt,b(1,j),0)
10    continue
```

computes $\mathbf{A}^{-1}\mathbf{B}$, storing the result in **B**.

Subroutine BCF (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCFLIN BCMET	Compute user-written boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
* IHSTAG	Flag for constant stagnation enthalpy option.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCF computes coefficients and source terms for user-written boundary conditions of the form $\Delta F = 0$, $F = f$, $\partial F/\partial \phi = f$, and $\nabla F \cdot \vec{n} = f$. The values of F and its derivatives with respect to the dependent variables must be supplied by the user-written subroutine BCFLIN. The linearized equations for these types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections expand these generalized equations in detail.[7]

[7] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

No Change From Initial Conditions, $\Delta F = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial\hat{\mathbf{Q}} = J\partial g/\partial\mathbf{Q}$, we get simply

$$J_{i,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = 0$$

Specified Value, $F = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = f^{n+1}_{i,j,k} - F^n_{i,j,k}$$

Specified Two-Point Gradient in Coordinate Direction, $\partial F/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\begin{aligned}
&-J_{1,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+J_{2,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} = \\
&\qquad (\Delta\xi)f^{n+1}_{1,j,k} + F^n_{1,j,k} - F^n_{2,j,k}
\end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned}
&-J_{N_1-1,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+J_{N_1,j,k}\left[\frac{\partial F}{\partial\rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&\qquad (\Delta\xi)f^{n+1}_{N_1,j,k} + F^n_{N_1-1,j,k} - F^n_{N_1,j,k}
\end{aligned}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Gradient in Coordinate Direction, $\partial F/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k} = \\
&\qquad 2(\Delta\xi)f^{n+1}_{1,j,k} + 3F^n_{1,j,k} - 4F^n_{2,j,k} + F^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k} \\
&- 4J_{N_1-1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+ 3J_{N_1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&\qquad 2(\Delta\xi)f^{n+1}_{N_1,j,k} - F^n_{N_1-2,j,k} + 4F^n_{N_1-1,j,k} - 3F^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Gradient in Normal Direction, $\nabla F \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- J_{1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+ J_{2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta F^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta F^n_{1,j,k}\right] \\
&\qquad + F^n_{1,j,k} - F^n_{2,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+ J_{N_1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta F^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta F^n_{N_1,j,k}\right] \\
&\quad + F^n_{N_1-1,j,k} - F^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Gradient in Normal Direction, $\nabla F \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta F^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta F^n_{1,j,k}\right] \\
&\quad + 3F^n_{1,j,k} - 4F^n_{2,j,k} + F^n_{3,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k} \\
&- 4J_{N_1-1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+ 3J_{N_1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta F^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta F^n_{N_1,j,k}\right] \\
&\quad - F^n_{N_1-2,j,k} + 4F^n_{N_1-1,j,k} - 3F^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$
\begin{aligned}
& J_{1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
& - 2J_{2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} \\
& + J_{3,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k} = \\
& - F^n_{1,j,k} + 2F^n_{2,j,k} - F^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
& J_{N_1-2,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k} \\
& - 2J_{N_1-1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
& + J_{N_1,j,k}\left[\frac{\partial F}{\partial \rho}\Delta\hat{\rho} + \frac{\partial F}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial F}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial F}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial F}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
& - F^n_{N_1-2,j,k} + 2F^n_{N_1-1,j,k} - F^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent user-written boundary condition is specified.

Subroutine BCFLIN (IBC,IEQ,IBOUND,IMIN,IMAX,F,DFDRHO,DFDRU,DFDRV,DFDRW, DFDET,FBCMA,FBCMB,FBCPA,FBCPB,FBC)		
Called by	Calls	Purpose
BCF		User-supplied routine for linearization of user-supplied boundary conditions.

Input

IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC	Mean flow boundary condition types for current sweep direction, specified as IBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
N1P, N2P	Parameters specifying the dimension size in the ξ and η directions.

Output

DFDRHO, DFDRU, DFDRV, DFDRW, DFDET	Three-element arrays, specified as DFDRHO(IW), etc., giving the values of $\partial F/\partial \rho$, $\partial F/\partial(\rho u)$, $\partial F/\partial(\rho v)$, $\partial F/\partial(\rho w)$, and $\partial F/\partial E_T$.
F	A three-element array specified as F(IW) giving the value of the function F at the boundary (IW = 1), at the first point away from the boundary (IW = 2), and at the second point away from the boundary (IW = 3). Values at IW = 3 are not needed for boundary condition types 91, 92, or -92. Values at IW = 2 are not needed for boundary condition type 91.
FBC	Boundary condition values for current sweep direction, specified as FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries. This is only needed if values for GBC1, GBC2, or GBC3, or FBC1, FBC2, or FBC3, are not specified in the input namelist BC.
FBCMA, FBCPA	Boundary condition values on the boundary, at the grid points "left" and "right" of the current boundary point, in the first non-sweep direction. These are only needed for boundary condition types ± 93.
FBCMB, FBCPB	Boundary condition values on the boundary, at the grid points "left" and "right" of the current boundary point, in the second non-sweep direction. These are only needed for boundary condition types ± 93.

Description

Subroutine BCFLIN is a user-written routine used in conjunction with subroutine BCF for user-written boundary conditions of the form $\Delta F = 0$, $F = f$, $\partial F / \partial \phi = f$, and $\nabla F \cdot \vec{n} = f$. BCFLIN supplies the values of F and its derivatives with respect to the dependent variables, which are required for writing the linearized form of the boundary condition.

The version of BCFLIN supplied with *Proteus* makes BCF equivalent to BCTEMP, except for the total temperature options in BCTEMP. Thus $F = T$, $\partial F / \partial \rho = \partial T / \partial \rho$, etc., where T and its derivatives with respect to the dependent variables are computed using the perfect gas equation of state. (See Section 4.3 of Volume 1.) This version of BCFLIN is intended as an example for use in coding boundary conditions not already available.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The capability of specifying FBC as an output variable may be useful in writing time-dependent boundary conditions. It also may be used when specifying boundary conditions involving derivatives in one of the non-sweep directions. In this case, the derivatives in the non-sweep directions may be lagged one time step and treated as source terms.

Subroutine BCGEN		
Called by	Calls	Purpose
BVUP EXEC	BCDENS BCF BCNVEL BCPRES BCQ BCTEMP BCUVEL BCVVEL BCWVEL BC1VEL BC2VEL BC3VEL BLKOUT ISRCHEQ	Manage computation of boundary conditions.

Input

* FBC1, FBC2, FBC3	Point-by-point mean flow boundary condition values for the ξ, η, and ζ directions.
* IBC1, IBC2, IBC3	Point-by-point mean flow boundary condition types for the ξ, η, and ζ directions.
* IDEBUG	Debug flags.
* IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
ISWEEP	Current ADI sweep number.
IT	Current time step number n.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
NBC	Dimensioning parameter specifying number of boundary conditions per equation.
NEQ	Number of coupled equations being solved, N_{eq}.
* NOUT	Unit number for standard output.
NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
NVD, NPTSD	Leading two dimensions for the arrays A, B, C, S, METX, METY, METZ, and METT.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

Output

IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and $J = 1$ or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number, from 1 to N_{eq}.

IMIN, IMAX	Minimum and maximum indices in the sweep direction.

Description

Subroutine BCGEN manages the computation of coefficients and source terms for the mean flow boundary conditions. It first loads the NEQ boundary condition types and values from the input arrays IBC1 and FBC1, IBC2 and FBC2, or IBC3 and FBC3, depending on the ADI sweep, into the arrays IBC and FBC. This was done so that the BC routines could be non-sweep dependent. Next the coefficient submatrices and source term subvectors at the two boundaries in the current sweep direction are initialized to zero. And finally, the appropriate BC routine is called, depending on the input boundary condition type, for each of the NEQ boundary conditions at each boundary in the sweep direction.

Remarks

1. An error message is generated and execution is stopped if the boundary condition type is less than 0 or greater than 99.
2. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.
3. This subroutine generates the output for the IDEBUG(3) option.

Subroutine BCGRAD (F,I,DFD1,DFD2,DFD3)		
Called by	Calls	Purpose
BCDENS BCF BCPRES BCQ BCTEMP BCUVEL BCVVEL BCWVEL		Compute gradients with respect to ξ, η, and ζ.

Input

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
F	A three-dimensional array, specified as F(I,J,K), containing the function f whose gradient is to be computed. The subscripts I, J, and K run from 1 to N_1, N_2, and N_3, respectively.
I	Current grid point index in the current sweep direction.
ISWEEP	Current ADI sweep number.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

Output

DFD1, DFD2, DFD3	First derivatives of f with respect to ξ, η, and ζ.

Description

Subroutine BCGRAD computes first derivatives of the function f, with respect to ξ, η, and ζ, at the current grid point in the ADI sweep direction. At interior points, the centered difference formula presented in Section 5.0 of Volume 1 is used. For derivatives with respect to ξ,

$$\left(\frac{\partial f}{\partial \xi}\right)_{i,j,k} \simeq \frac{f_{i+1,j,k} - f_{i-1,j,k}}{\Delta\xi}$$

An analogous formula is used for η and ζ derivatives.

At boundary points three-point one-sided formulas are used.

$$\left(\frac{\partial f}{\partial \xi}\right)_{1,j,k} \simeq \frac{1}{2\Delta\xi}(-3f_{1,j,k} + 4f_{2,j,k} - f_{3,j,k})$$

$$\left(\frac{\partial f}{\partial \xi}\right)_{N_1,j,k} \simeq \frac{1}{2\Delta\xi}(f_{N_1-2,j,k} - 4f_{N_1-1,j,k} + 3f_{N_1,j,k})$$

Again, analogous formulas are used for η and ζ derivatives.

Subroutine BCIMET (I,XXI,XETA,XZETA,YXI,YETA,YZETA,ZXI,ZETA,ZZETA)		
Called by	Calls	Purpose
BC1VEL BC2VEL BC3VEL		Compute inverse metrics at a point in the current sweep direction.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
I	Current grid point index in the current sweep direction.
ISWEEP	Current ADI sweep number.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

XXI, XETA, XZETA	Derivatives of x with respect to ξ, η, and ζ.
YXI, YETA, YZETA	Derivatives of y with respect to ξ, η, and ζ.
ZXI, ZETA, ZZETA	Derivatives of z with respect to ξ, η, and ζ.

Description

Subroutine BCIMET computes the inverse metrics using the following formulas:

$$x_\xi = \frac{1}{J}(\eta_y\zeta_z - \eta_z\zeta_y)$$

$$y_\xi = \frac{1}{J}(\eta_z\zeta_x - \eta_x\zeta_z)$$

$$z_\xi = \frac{1}{J}(\eta_x\zeta_y - \eta_y\zeta_x)$$

$$x_\eta = \frac{1}{J}(\xi_z\zeta_y - \xi_y\zeta_z)$$

$$y_\eta = \frac{1}{J}(\xi_x\zeta_z - \xi_z\zeta_x)$$

$$z_\eta = \frac{1}{J}(\xi_y\zeta_x - \xi_x\zeta_y)$$

$$x_\zeta = \frac{1}{J}(\xi_y\eta_z - \xi_z\eta_y)$$

$$y_\zeta = \frac{1}{J}(\xi_z\eta_x - \xi_x\eta_z)$$

$$z_\zeta = \frac{1}{J}(\xi_x\eta_y - \xi_y\eta_x)$$

Subroutine BCMET (I,FM0,FM1,FM2,FM3)		
Called by	Calls	Purpose
BCDENS BCF BCNVEL BCPRES BCQ BCTEMP BCUVEL BCVVEL BCWVEL BC1VEL BC2VEL BC3VEL		Compute various metric functions for normal gradient boundary conditions.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
I	Current grid point index in the current sweep direction.
ISWEEP	Current ADI sweep number.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

FM0, FM1, FM2, FM3	Various metric functions used for normal derivative boundary conditions.

Description

Subroutine BCMET computes metric functions used for normal gradient boundary conditions. For the first ADI sweep,

$$
\begin{aligned}
FM0 &= \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \\
FM1 &= 0 \\
FM2 &= \xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z \\
FM3 &= \xi_x \zeta_x + \xi_y \zeta_y + \xi_z \zeta_z
\end{aligned}
$$

For the second sweep,

$$
\begin{aligned}
FM0 &= \sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2} \\
FM1 &= \xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z \\
FM2 &= 0 \\
FM3 &= \eta_x \zeta_x + \eta_y \zeta_y + \eta_z \zeta_z
\end{aligned}
$$

And for the third sweep,

$$
\begin{aligned}
\text{FM0} &= \sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2} \\
\text{FM1} &= \xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z \\
\text{FM2} &= \eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z \\
\text{FM3} &= 0
\end{aligned}
$$

Subroutine BCNVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCMET BCVN	Compute normal direction velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
METX, METY, METZ	Derivatives of sweep direction computational coordinate with respect to x, y, and z.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW	Array indices associated with the dependent variables ρ, ρu, ρv, and ρw.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCNVEL computes coefficients and source terms for normal direction velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular normal direction velocity boundary conditions in *Proteus*.[8]

[8] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the

Specified Normal Velocity, $V_n = f$

The normal velocity is defined as

$$V_n = \vec{V} \bullet \vec{n}$$

where $\vec{n}$ is the unit vector normal to the boundary. For a ξ boundary,

$$\vec{n} = \frac{\nabla \xi}{|\nabla \xi|} = \frac{1}{m} \xi_x \vec{i} + \frac{1}{m} \xi_y \vec{j} + \frac{1}{m} \xi_z \vec{k}$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

Therefore, for a ξ boundary,

$$V_n = \frac{1}{m} (\xi_x u + \xi_y v + \xi_z w)$$

Note that the unit vector $\vec{n}$ is in the direction of increasing ξ. Therefore V_n is positive in the direction of increasing ξ. Thus, a positive V_n at $\xi = 0$ implies flow into the computational domain, and a positive V_n at $\xi = 1$ implies flow out of the computational domain.

Similarly, for an η boundary,

$$V_n = \frac{1}{m} (\eta_x u + \eta_y v + \eta_z w)$$

where

$$m = \sqrt{\eta_x^{\,2} + \eta_y^{\,2} + \eta_z^{\,2}}$$

and V_n is positive in the direction of increasing η.

And, for a ζ boundary,

$$V_n = \frac{1}{m} (\zeta_x u + \zeta_y v + \zeta_z w)$$

where

$$m = \sqrt{\zeta_x^{\,2} + \zeta_y^{\,2} + \zeta_z^{\,2}}$$

and V_n is positive in the direction of increasing ζ.

Applying equation (6.5) of Volume 1, the linearized boundary condition at a ξ boundary becomes

$$\frac{J_{i,j,k}}{m_{i,j,k}} \left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho} \Delta\hat{\rho} + \frac{\xi_x}{\rho} \Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho} \Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho} \Delta(\widehat{\rho w}) \right]^n_{i,j,k} = f^{n+1}_{i,j,k} - (V_n)^n_{i,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

Specified Two-Point Normal Velocity Gradient in Coordinate Direction, $\partial V_n/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{1,j,k}^{n} \\
&+\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{2,j,k}^{n} = \\
&\qquad (\Delta\xi) f_{1,j,k}^{n+1} + (V_n)_{1,j,k}^{n} - (V_n)_{2,j,k}^{n}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{N_1-1,j,k}^{n} \\
&+\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{N_1,j,k}^{n} = \\
&\qquad (\Delta\xi) f_{N_1,j,k}^{n+1} + (V_n)_{N_1-1,j,k}^{n} - (V_n)_{N_1,j,k}^{n}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Normal Velocity Gradient in Coordinate Direction, $\partial V_n/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&-3\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{1,j,k}^{n} \\
&+4\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{2,j,k}^{n} \\
&-\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{3,j,k}^{n} = \\
&\qquad 2(\Delta\xi) f_{1,j,k}^{n+1} + 3(V_n)_{1,j,k}^{n} - 4(V_n)_{2,j,k}^{n} + (V_n)_{3,j,k}^{n}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{N_1-2,j,k}^{n} \\
&-4\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{N_1-1,j,k}^{n} \\
&+3\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\widehat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\widehat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\widehat{\rho w})\right]_{N_1,j,k}^{n} = \\
&\qquad 2(\Delta\xi) f_{N_1,j,k}^{n+1} - (V_n)_{N_1-2,j,k}^{n} + 4(V_n)_{N_1-1,j,k}^{n} - 3(V_n)_{N_1,j,k}^{n}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Normal Velocity Gradient in Normal Direction, $\nabla V_n \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} \\
&+\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta(V_n)^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta(V_n)^n_{1,j,k}\right] \\
&\qquad + (V_n)^n_{1,j,k} - (V_n)^n_{2,j,k}
\end{aligned}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} \\
&+\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta(V_n)^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta(V_n)^n_{N_1,j,k}\right] \\
&\qquad + (V_n)^n_{N_1-1,j,k} - (V_n)^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Normal Velocity Gradient in Normal Direction, $\nabla V_n \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&-3\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} \\
&+4\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} \\
&-\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta(V_n)^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta(V_n)^n_{1,j,k}\right] \\
&\qquad + 3(V_n)^n_{1,j,k} - 4(V_n)^n_{2,j,k} + (V_n)^n_{3,j,k}
\end{aligned}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k}$$
$$-4\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}$$
$$+3\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta(V_n)^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta(V_n)^n_{N_1,j,k}\right]$$
$$-(V_n)^n_{N_1-2,j,k} + 4(V_n)^n_{N_1-1,j,k} - 3(V_n)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of Normal Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k}$$
$$-2\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k}$$
$$+\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} =$$
$$-(V_n)^n_{1,j,k} + 2(V_n)^n_{2,j,k} - (V_n)^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k}$$
$$-2\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}$$
$$+\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{\xi_x u + \xi_y v + \xi_z w}{\rho}\Delta\hat{\rho} + \frac{\xi_x}{\rho}\Delta(\hat{\rho u}) + \frac{\xi_y}{\rho}\Delta(\hat{\rho v}) + \frac{\xi_z}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$-(V_n)^n_{N_1-2,j,k} + 2(V_n)^n_{N_1-1,j,k} - (V_n)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.

2. An error message is generated and execution is stopped if a non-existent normal direction velocity boundary condition is specified.

Subroutine BCPRES (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute pressure boundary conditions.

Input

CP, CV	Specific heats c_p and c_v at time level n.
DEL	Computational grid spacing in sweep direction.
DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial \rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
DTDRHO, DTDRU, DTDRV, DTDRW, DTDET	Derivatives $\partial T/\partial \rho$, $\partial T/\partial(\rho u)$, $\partial T/\partial(\rho v)$, $\partial T/\partial(\rho w)$, and $\partial T/\partial E_T$.
GC	Proportionality factor g_c in Newton's second law.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
* IHSTAG	Flag for constant stagnation enthalpy option.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
P, T	Static pressure p and temperature T at time level n.
PR	Reference pressure p_r.
RGAS	Gas constant R.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.
* RHOR, UR	Reference density ρ_r and velocity u_r.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCPRES computes coefficients and source terms for pressure boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular pressure boundary conditions in *Proteus*.[9]

No Change From Initial Conditions, $\Delta p = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial \hat{\mathbf{Q}} = J\partial g/\partial \mathbf{Q}$, we get simply

$$J_{i,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = 0$$

The derivatives $\partial p/\partial \rho$, $\partial p/\partial(\rho u)$, etc., depend on the equation of state. They are defined for a perfect gas in Section 4.3 of Volume 1.

Specified Static Pressure, $p = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = f^{n+1}_{i,j,k}\frac{p_r g_c}{\rho_r u_r^2} - p^n_{i,j,k}$$

Specified Two-Point Pressure Gradient in Coordinate Direction, $\partial p/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\begin{aligned}
&-J_{1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+J_{2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} = \\
&(\Delta\xi) f^{n+1}_{1,j,k}\frac{p_r g_c}{\rho_r u_r^2} + p^n_{1,j,k} - p^n_{2,j,k}
\end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned}
&-J_{N_1-1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+J_{N_1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&(\Delta\xi) f^{n+1}_{N_1,j,k}\frac{p_r g_c}{\rho_r u_r^2} + p^n_{N_1-1,j,k} - p^n_{N_1,j,k}
\end{aligned}$$

[9] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Pressure Gradient in Coordinate Direction, $\partial p/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{3,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{1,j,k}\frac{p_r g_c}{\rho_r u_r^2} + 3p^n_{1,j,k} - 4p^n_{2,j,k} + p^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-2,j,k} \\
&- 4J_{N_1-1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-1,j,k} \\
&+ 3J_{N_1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{N_1,j,k}\frac{p_r g_c}{\rho_r u_r^2} - p^n_{N_1-2,j,k} + 4p^n_{N_1-1,j,k} - 3p^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Pressure Gradient in Normal Direction, $\nabla p \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- J_{1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{1,j,k} \\
&+ J_{2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T \right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k}\frac{p_r g_c}{\rho_r u_r^2} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta p^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta p^n_{1,j,k} \right] \\
&\quad + p^n_{1,j,k} - p^n_{2,j,k}
\end{aligned}
$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+ J_{N_1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}\frac{p_r g_c}{\rho_r u_r^2} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta p^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta p^n_{N_1,j,k}\right] \\
&\qquad + p^n_{N_1-1,j,k} - p^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Pressure Gradient in Normal Direction, $\nabla p \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k}\frac{p_r g_c}{\rho_r u_r^2} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta p^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta p^n_{1,j,k}\right] \\
&\qquad + 3p^n_{1,j,k} - 4p^n_{2,j,k} + p^n_{3,j,k}
\end{aligned}
$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k}\\
&-4J_{N_1-1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k}\\
&+3J_{N_1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k}=\\
&\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}\frac{p_r g_c}{\rho_r u_r^2}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta p^n_{N_1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta p^n_{N_1,j,k}\right]\\
&\qquad -p^n_{N_1-2,j,k}+4p^n_{N_1-1,j,k}-3p^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of Static Pressure

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$
\begin{aligned}
&J_{1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k}\\
&-2J_{2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k}\\
&+J_{3,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k}=\\
&\qquad -p^n_{1,j,k}+2p^n_{2,j,k}-p^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k}\\
&-2J_{N_1-1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k}\\
&+J_{N_1,j,k}\left[\frac{\partial p}{\partial \rho}\Delta\hat{\rho}+\frac{\partial p}{\partial(\rho u)}\Delta(\hat{\rho u})+\frac{\partial p}{\partial(\rho v)}\Delta(\hat{\rho v})+\frac{\partial p}{\partial(\rho w)}\Delta(\hat{\rho w})+\frac{\partial p}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k}=\\
&\qquad -p^n_{N_1-2,j,k}+2p^n_{N_1-1,j,k}-p^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

No Change From Initial Conditions for Total Pressure, $\Delta p_T = 0$

The total pressure is defined as

$$p_T = p\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}}$$

Applying equation (6.3) of Volume 1, we get

$$J_{i,j,k}\left[\frac{\partial p_T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p_T}{\partial(\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial p_T}{\partial(\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial p_T}{\partial(\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial p_T}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = 0$$

where

$$\frac{\partial p_T}{\partial \rho} = \frac{\partial p}{\partial \rho}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} + p\frac{\gamma}{2}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \frac{\partial M^2}{\partial \rho}$$

$$\frac{\partial p_T}{\partial(\rho u)} = \frac{\partial p}{\partial(\rho u)}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} + p\frac{\gamma}{2}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \frac{\partial M^2}{\partial(\rho u)}$$

$$\frac{\partial p_T}{\partial(\rho v)} = \frac{\partial p}{\partial(\rho v)}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} + p\frac{\gamma}{2}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \frac{\partial M^2}{\partial(\rho v)}$$

$$\frac{\partial p_T}{\partial(\rho w)} = \frac{\partial p}{\partial(\rho w)}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} + p\frac{\gamma}{2}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \frac{\partial M^2}{\partial(\rho w)}$$

$$\frac{\partial p_T}{\partial E_T} = \frac{\partial p}{\partial E_T}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} + p\frac{\gamma}{2}\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \frac{\partial M^2}{\partial E_T}$$

The Mach number is defined by

$$M^2 = \frac{u^2 + v^2 + w^2}{\gamma R T} = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\gamma R \rho^2 T}$$

The derivatives $\partial M^2/\partial\rho$, etc., can then be derived as

$$\frac{\partial M^2}{\partial \rho} = -M^2\left(\frac{2}{\rho} + \frac{1}{T}\frac{\partial T}{\partial \rho}\right)$$

$$\frac{\partial M^2}{\partial(\rho u)} = \frac{2u}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial(\rho u)}$$

$$\frac{\partial M^2}{\partial(\rho v)} = \frac{2v}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial(\rho v)}$$

$$\frac{\partial M^2}{\partial(\rho w)} = \frac{2w}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial(\rho w)}$$

$$\frac{\partial M^2}{\partial E_T} = -\frac{M^2}{T}\frac{\partial T}{\partial E_T}$$

Specified Total Pressure, $p_T = f$

Applying equation (6.5) of Volume 1, we get

$$J_{i,j,k}\left[\frac{\partial p_T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial p_T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial p_T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial p_T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial p_T}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} =$$

$$f^{n+1}_{i,j,k}\frac{p_r g_c}{\rho_r u_r^2} - p^n_{i,j,k}\left[\left(1 + \frac{\gamma - 1}{2}M^2\right)^{\frac{\gamma}{\gamma - 1}}\right]^n_{i,j,k}$$

where p_T, $\partial p_T/\partial\rho$, etc., are defined above as part of the description of the $\Delta p_T = 0$ boundary condition.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent pressure boundary condition is specified.
3. The multiplying factor $p_r g_c/\rho_r u_r^2$ that appears with specified values of pressure and pressure gradients is necessary because input values of pressure are nondimensionalized by the reference pressure $p_r = \rho_r R T_r/g_c$, while internal to the *Proteus* code itself pressure is nondimensionalized by the normalizing pressure $p_n = \rho_r u_r^2$. (See Section 3.1.1 of Volume 2.)

Subroutine BCQ (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute conservation variable boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations.
* NOUT	Unit number for standard output.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCQ computes coefficients and source terms for conservation variable boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular conservation variable boundary conditions in *Proteus*.[10]

---

[10] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the

No Change From Initial Conditions, $\Delta Q = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial \hat{\mathbf{Q}} = J\partial g/\partial \mathbf{Q}$, we get simply

$$J_{i,j,k}\Delta\hat{Q}^n_{i,j,k} = 0$$

where $\hat{Q}$ is the element of $\hat{\mathbf{Q}}$ for which this boundary condition is to be applied.

Specified Conservation Variable, $Q = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\Delta\hat{Q}^n_{i,j,k} = f^{n+1}_{i,j,k} - Q^n_{i,j,k}$$

Specified Two-Point Conservation Variable Gradient in Coordinate Direction, $\partial Q/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j,k}\Delta\hat{Q}^n_{1,j,k} + J_{2,j,k}\Delta\hat{Q}^n_{2,j,k} = (\Delta\xi)f^{n+1}_{1,j,k} + Q^n_{1,j,k} - Q^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j,k}\Delta\hat{Q}^n_{N_1-1,j,k} + J_{N_1,j,k}\Delta\hat{Q}^n_{N_1,j,k} = (\Delta\xi)f^{n+1}_{N_1,j,k} + Q^n_{N_1-1,j,k} - Q^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Conservation Variable Gradient in Coordinate Direction, $\partial Q/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j,k}\Delta\hat{Q}^n_{1,j,k} + 4J_{2,j,k}\Delta\hat{Q}^n_{2,j,k} - J_{3,j,k}\Delta\hat{Q}^n_{3,j,k} =$$
$$2(\Delta\xi)f^{n+1}_{1,j,k} + 3Q^n_{1,j,k} - 4Q^n_{2,j,k} + Q^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\Delta\hat{Q}^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\Delta\hat{Q}^n_{N_1-1,j,k} + 3J_{N_1,j,k}\Delta\hat{Q}^n_{N_1,j,k} =$$
$$2(\Delta\xi)f^{n+1}_{N_1,j,k} - Q^n_{N_1-2,j,k} + 4Q^n_{N_1-1,j,k} - 3Q^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Conservation Variable Gradient in Normal Direction, $\nabla Q \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

$$
\begin{aligned}
&- J_{1,j,k}\,\Delta\hat{Q}^n_{1,j,k} + J_{2,j,k}\,\Delta\hat{Q}^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta Q^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta Q^n_{1,j,k}\right] \\
&+ Q^n_{1,j,k} - Q^n_{2,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\,\Delta\hat{Q}^n_{N_1-1,j,k} + J_{N_1,j,k}\,\Delta\hat{Q}^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta Q^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta Q^n_{N_1,j,k}\right] \\
&+ Q^n_{N_1-1,j,k} - Q^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Conservation Variable Gradient in Normal Direction, $\nabla Q \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\,\Delta\hat{Q}^n_{1,j,k} + 4J_{2,j,k}\,\Delta\hat{Q}^n_{2,j,k} - J_{3,j,k}\,\Delta\hat{Q}^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta Q^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta Q^n_{1,j,k}\right] \\
&+ 3Q^n_{1,j,k} - 4Q^n_{2,j,k} + Q^n_{3,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\,\Delta\hat{Q}^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\,\Delta\hat{Q}^n_{N_1-1,j,k} + 3J_{N_1,j,k}\,\Delta\hat{Q}^n_{N_1,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta Q^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta Q^n_{N_1,j,k}\right] \\
&- Q^n_{N_1-2,j,k} + 4Q^n_{N_1-1,j,k} - 3Q^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of Conservation Variable

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$
J_{1,j,k}\,\Delta\hat{Q}^n_{1,j,k} - 2J_{2,j,k}\,\Delta\hat{Q}^n_{2,j,k} + J_{3,j,k}\,\Delta\hat{Q}^n_{3,j,k} = - Q^n_{1,j,k} + 2Q^n_{2,j,k} - Q^n_{3,j,k}
$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\Delta\hat{Q}^n_{N_1-2,j,k} - 2J_{N_1-1,j,k}\Delta\hat{Q}^n_{N_1-1,j,k} - J_{N_1,j,k}\Delta\hat{Q}^n_{N_1,j,k} = -Q^n_{N_1-2,j,k} + 2Q^n_{N_1-1,j,k} - Q^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent conservation variable boundary condition is specified.

Subroutine BCSET		
Called by	Calls	Purpose
MAIN		Set various boundary condition parameters and flags.

Input

* GBC1, GBC2, GBC3	Surface mean flow boundary condition values for the ξ, η, and ζ directions.
* GBCT1, GBCT2, GBCT3	Surface k-ε boundary condition values for the ξ, η, and ζ directions.
* GTBC1, GTBC2, GTBC3	Time-dependent surface mean flow boundary condition values for the ξ, η, and ζ directions.
* IBC1, IBC2, IBC3	Point-by-point mean flow boundary condition types for the ξ, η, and ζ directions (if set in input.)
* IHSTAG	Flag for constant stagnation enthalpy option.
ITDBC	Flag for time-dependent mean flow boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used.
* ITURB	Flag for turbulent flow option.
* JBC1, JBC2, JBC3	Surface mean flow boundary condition types for the ξ, η, and ζ directions.
* JBCT1, JBCT2, JBCT3	Surface k-ε boundary condition types for the ξ, η, and ζ directions.
* JTBC1, JTBC2, JTBC3	Flags for type of time dependency for mean flow boundary conditions in the ξ, η, and ζ directions.
* KBC1, KBC2 KBC3	Boundary types for the ξ, η, and ζ directions.
NBC	Dimensioning parameter specifying number of boundary conditions per equation.
NEQ	Number of coupled equations being solved, N_{eq}.
* NOUT	Unit number for standard output.
* NTBC	Number of values in tables for general unsteady boundary conditions.
* NTBCA	Time levels at which general unsteady boundary conditions are specified.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

Output

FBC1, FBC2, FBC3	Point-by-point mean flow boundary condition values for the ξ, η, and ζ directions.
FBCT1, FBCT2, FBCT3	Point-by-point k-ε boundary condition values for the ξ, η, and ζ directions.
IBCELM	Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the ξ and/or η directions; 0 if elimination is not necessary, 1 if it is.

IBC1, IBC2, IBC3	Point-by-point mean flow boundary condition types for the ξ, η, and ζ directions.
IBCT1, IBCT2, IBCT3	Point-by-point k-ε boundary condition types for the ξ, η, and ζ directions.
IBVUP	Flags for updating boundary values from first two sweeps after third sweep; 0 if updating is not necessary, 1 if it is.
JBC1, JBC2, JBC3	Surface mean flow boundary condition types for the ξ, η, and ζ directions (if using the KBC meta flags.)
KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.

Description

Subroutine BCSET sets various boundary condition parameters and flags. It first sets NPT1, NPT2, and NPT3, the number of grid points in each ADI sweep direction to be used in computing coefficients and source terms. For spatially periodic boundary conditions in the ξ direction, NPT1 = N1 + 1. Similarly, for spatially periodic boundary conditions in the η direction, NPT2 = N2 + 1. And, for spatially periodic boundary conditions in the ζ direction, NPT3 = N3 + 1. This is done in order to use central differences at the periodic boundary. (See Section 7.2.2 of Volume 1.)

Next, if the boundary types are being specified using the KBC meta flags, the appropriate mean flow surface boundary condition types are loaded into the JBC arrays. Special flags are set if spatially periodic boundary conditions are being used. Then, unless the mean flow boundary conditions are being specified point-by-point using the IBC and FBC parameters, the appropriate IBC and FBC parameters are loaded with the JBC and GBC values.

If three-point gradient or extrapolation mean flow boundary conditions are being used, a flag is set for eliminating the resulting off-diagonal coefficient submatrix. If gradient (two-point or three-point) or extrapolation mean flow boundary conditions are used during the first or second sweep, a flag is set for updating the ξ and η boundary values after the third sweep.

Next, for turbulent flow using the k-ε model, if the k-ε boundary conditions are being specified using the JBCT and GBCT parameters, the appropriate point-by-point boundary condition types and values (the IBCT and FBCT parameters) are loaded with the JBCT and GBCT values.

And finally, the input boundary condition parameters are then written to the standard output file.

Remarks

1. An error message is generated and execution is stopped if an invalid boundary type is specified with the KBC meta flags.

Subroutine BCTEMP (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute temperature boundary conditions.

Input

CP, CV	Specific heats c_p and c_v at time level n.
DEL	Computational grid spacing in sweep direction.
DTDRHO, DTDRU, DTDRV, DTDRW, DTDET	Derivatives $\partial T/\partial\rho$, $\partial T/\partial(\rho u)$, $\partial T/\partial(\rho v)$, $\partial T/\partial(\rho w)$, and $\partial T/\partial E_T$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
* IHSTAG	Flag for constant stagnation enthalpy option.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
P, T	Static pressure p and temperature T at time level n.
RGAS	Gas constant R.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCTEMP computes coefficients and source terms for temperature boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0

of Volume 1. The following sections apply these generalized equations to the particular temperature boundary conditions in *Proteus*.[11]

No Change From Initial Conditions, $\Delta T = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial \hat{\mathbf{Q}} = J \partial g/\partial \mathbf{Q}$, we get simply

$$J_{i,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j,k}^{n} = 0$$

The derivatives $\partial T/\partial \rho$, $\partial T/\partial(\rho u)$, etc., depend on the equation of state. They are defined for a perfect gas in Section 4.3 of Volume 1.

Specified Static Temperature, $T = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j,k}^{n} = f_{i,j,k}^{n+1} - T_{i,j,k}^{n}$$

Specified Two-Point Temperature Gradient in Coordinate Direction, $\partial T/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\begin{aligned} & -J_{1,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j,k}^{n} \\ & + J_{2,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j,k}^{n} = \\ & \qquad (\Delta \xi) f_{1,j,k}^{n+1} + T_{1,j,k}^{n} - T_{2,j,k}^{n} \end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned} & -J_{N_1-1,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1-1,j,k}^{n} \\ & + J_{N_1,j,k} \left[\frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho u}) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho v}) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho w}) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j,k}^{n} = \\ & \qquad (\Delta \xi) f_{N_1,j,k}^{n+1} + T_{N_1-1,j,k}^{n} - T_{N_1,j,k}^{n} \end{aligned}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Temperature Gradient in Coordinate Direction, $\partial T/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

[11] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{3,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{1,j,k} + 3T^n_{1,j,k} - 4T^n_{2,j,k} + T^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-2,j,k} \\
&- 4J_{N_1-1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-1,j,k} \\
&+ 3J_{N_1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{N_1,j,k} - T^n_{N_1-2,j,k} + 4T^n_{N_1-1,j,k} - 3T^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point Temperature Gradient in Normal Direction, $\nabla T \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- J_{1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{1,j,k} \\
&+ J_{2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta T^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta T^n_{1,j,k} \right] \\
&\qquad + T^n_{1,j,k} - T^n_{2,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{{\xi_x}^2 + {\xi_y}^2 + {\xi_z}^2}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-1,j,k} \\
&+ J_{N_1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta T^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta T^n_{N_1,j,k} \right] \\
&\qquad + T^n_{N_1-1,j,k} - T^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point Temperature Gradient in Normal Direction, $\nabla T \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{1,j,k} \\
&+ 4J_{2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{2,j,k} \\
&- J_{3,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta T^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta T^n_{1,j,k} \right] \\
&\qquad + 3T^n_{1,j,k} - 4T^n_{2,j,k} + T^n_{3,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-2,j,k} \\
&- 4J_{N_1-1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1-1,j,k} \\
&+ 3J_{N_1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T \right]^n_{N_1,j,k} = \\
&\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta T^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta T^n_{N_1,j,k} \right] \\
&\qquad - T^n_{N_1-2,j,k} + 4T^n_{N_1-1,j,k} - 3T^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of Static Temperature

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$\begin{aligned}
&J_{1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{1,j,k} \\
&- 2J_{2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{2,j,k} \\
&+ J_{3,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{3,j,k} = \\
&- T^n_{1,j,k} + 2T^n_{2,j,k} - T^n_{3,j,k}
\end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned}
&J_{N_1-2,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-2,j,k} \\
&- 2J_{N_1-1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1-1,j,k} \\
&+ J_{N_1,j,k}\left[\frac{\partial T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T}{\partial E_T}\Delta\hat{E}_T\right]^n_{N_1,j,k} = \\
&- T^n_{N_1-2,j,k} + 2T^n_{N_1-1,j,k} - T^n_{N_1,j,k}
\end{aligned}$$

Analogous equations can easily be written for the η and ζ boundaries.

No Change From Initial Conditions for Total Temperature, $\Delta T_T = 0$

The total temperature is defined as

$$T_T = T\left(1 + \frac{\gamma - 1}{2}M^2\right)$$

Applying equation (6.3) of Volume 1, we get

$$J_{i,j,k}\left[\frac{\partial T_T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T_T}{\partial(\rho u)}\Delta(\hat{\rho u}) + \frac{\partial T_T}{\partial(\rho v)}\Delta(\hat{\rho v}) + \frac{\partial T_T}{\partial(\rho w)}\Delta(\hat{\rho w}) + \frac{\partial T_T}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} = 0$$

where

$$\frac{\partial T_T}{\partial \rho} = \frac{\partial T}{\partial \rho}\left(1 + \frac{\gamma - 1}{2} M^2\right) + \frac{\gamma - 1}{2} T \frac{\partial M^2}{\partial \rho}$$

$$\frac{\partial T_T}{\partial (\rho u)} = \frac{\partial T}{\partial (\rho u)}\left(1 + \frac{\gamma - 1}{2} M^2\right) + \frac{\gamma - 1}{2} T \frac{\partial M^2}{\partial (\rho u)}$$

$$\frac{\partial T_T}{\partial (\rho v)} = \frac{\partial T}{\partial (\rho v)}\left(1 + \frac{\gamma - 1}{2} M^2\right) + \frac{\gamma - 1}{2} T \frac{\partial M^2}{\partial (\rho v)}$$

$$\frac{\partial T_T}{\partial (\rho w)} = \frac{\partial T}{\partial (\rho w)}\left(1 + \frac{\gamma - 1}{2} M^2\right) + \frac{\gamma - 1}{2} T \frac{\partial M^2}{\partial (\rho w)}$$

$$\frac{\partial T_T}{\partial E_T} = \frac{\partial T}{\partial E_T}\left(1 + \frac{\gamma - 1}{2} M^2\right) + \frac{\gamma - 1}{2} T \frac{\partial M^2}{\partial E_T}$$

The Mach number is defined by

$$M^2 = \frac{u^2 + v^2 + w^2}{\gamma R T} = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\gamma R \rho^2 T}$$

The derivatives $\partial M^2/\partial \rho$, etc., can then be derived as

$$\frac{\partial M^2}{\partial \rho} = -M^2\left(\frac{2}{\rho} + \frac{1}{T}\frac{\partial T}{\partial \rho}\right)$$

$$\frac{\partial M^2}{\partial (\rho u)} = \frac{2u}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial (\rho u)}$$

$$\frac{\partial M^2}{\partial (\rho v)} = \frac{2v}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial (\rho v)}$$

$$\frac{\partial M^2}{\partial (\rho w)} = \frac{2w}{\gamma p} - \frac{M^2}{T}\frac{\partial T}{\partial (\rho w)}$$

$$\frac{\partial M^2}{\partial E_T} = -\frac{M^2}{T}\frac{\partial T}{\partial E_T}$$

Specified Total Temperature, $T_T = f$

Applying equation (6.5) of Volume 1, we get

$$J_{i,j,k}\left[\frac{\partial T_T}{\partial \rho}\Delta\hat{\rho} + \frac{\partial T_T}{\partial (\rho u)}\Delta(\widehat{\rho u}) + \frac{\partial T_T}{\partial (\rho v)}\Delta(\widehat{\rho v}) + \frac{\partial T_T}{\partial (\rho w)}\Delta(\widehat{\rho w}) + \frac{\partial T_T}{\partial E_T}\Delta\hat{E}_T\right]^n_{i,j,k} =$$

$$f^{n+1}_{i,j,k} - T^n_{i,j,k}\left(1 + \frac{\gamma - 1}{2} M^2\right)^n_{i,j,k}$$

where T_T, $\partial T_T/\partial \rho$, etc., are defined above as part of the description of the $\Delta T_T = 0$ boundary condition.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.

2. An error message is generated and execution is stopped if a non-existent temperature boundary condition is specified.

Subroutine BCUVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute x-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU	Array indices associated with the dependent variables ρ and ρu.
RHO, U	Static density ρ and velocity u at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCUVEL computes coefficients and source terms for x-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular x-velocity boundary conditions in *Proteus*.[12]

[12] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

No Change From Initial Conditions, $\Delta u = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g / \partial \hat{\mathbf{Q}} = J \partial g / \partial \mathbf{Q}$, we get simply

$$J_{i,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{i,j,k} = 0$$

Specified x-Velocity, $u = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{i,j,k} = f^{n+1}_{i,j,k} - u^n_{i,j,k}$$

Specified Two-Point x-Velocity Gradient in Coordinate Direction, $\partial u / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{1,j,k} + J_{2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{2,j,k} = (\Delta\xi) f^{n+1}_{1,j,k} + u^n_{1,j,k} - u^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-1,j,k} + J_{N_1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1,j,k} = (\Delta\xi) f^{n+1}_{N_1,j,k} + u^n_{N_1-1,j,k} - u^n_{N_1,j,k}$$

Analogous equatio:: s can easily be written for the η and ζ boundaries.

Specified Three-Point x-Velocity Gradient in Coordinate Direction, $\partial u / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{1,j,k} + 4J_{2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{2,j,k}$$
$$-J_{3,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{3,j,k} = 2(\Delta\xi) f^{n+1}_{1,j,k} + 3u^n_{1,j,k} - 4u^n_{2,j,k} + u^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-1,j,k}$$
$$+ 3J_{N_1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1,j,k} = 2(\Delta\xi) f^{n+1}_{N_1,j,k} - u^n_{N_1-2,j,k} + 4u^n_{N_1-1,j,k} - 3u^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point x-Velocity Gradient in Normal Direction, $\nabla u \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$- J_{1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{1,j,k} + J_{2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{2,j,k} =$$

$$\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta u^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta u^n_{1,j,k}\right]$$
$$+ u^n_{1,j,k} - u^n_{2,j,k}$$

where

$$m = \sqrt{\xi_x^{\ 2} + \xi_y^{\ 2} + \xi_z^{\ 2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$- J_{N_1-1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-1,j,k} + J_{N_1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1,j,k} =$$

$$\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta u^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta u^n_{N_1,j,k}\right]$$
$$+ u^n_{N_1-1,j,k} - u^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point x-Velocity Gradient in Normal Direction, $\nabla u \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- 3J_{1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{1,j,k} + 4J_{2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{2,j,k}$$
$$- J_{3,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{3,j,k} =$$

$$\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta u^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta u^n_{1,j,k}\right]$$
$$+ 3u^n_{1,j,k} - 4u^n_{2,j,k} + u^n_{3,j,k}$$

where

$$m = \sqrt{\xi_x^{\ 2} + \xi_y^{\ 2} + \xi_z^{\ 2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-1,j,k}$$

$$+ 3J_{N_1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1,j,k} =$$

$$\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta u^n_{N_1,j,k} - \frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta u^n_{N_1,j,k}\right]$$

$$- u^n_{N_1-2,j,k} + 4u^n_{N_1-1,j,k} - 3u^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of x-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$J_{1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{1,j,k} - 2J_{2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{2,j,k}$$

$$+ J_{3,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{3,j,k} = - u^n_{1,j,k} + 2u^n_{2,j,k} - u^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-2,j,k} - 2J_{N_1-1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1-1,j,k}$$

$$+ J_{N_1,j,k}\left[-\frac{u}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho u})\right]^n_{N_1,j,k} = - u^n_{N_1-2,j,k} + 2u^n_{N_1-1,j,k} - u^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent x-velocity boundary condition is specified.

Subroutine BCVN (J1,J2,J3,VN)		
Called by	Calls	Purpose
BCNVEL		Compute velocity normal to a surface.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
ISWEEP	Current ADI sweep number.
J1, J2, J3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
U, V, W	Velocities u, v, and w, at time level n.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VN	Velocity normal to sweep direction surface.

Description

Subroutine BCVN computes the velocity normal to a surface in the current sweep direction. The normal velocity is defined as

$$V_n = \vec{V} \cdot \vec{n}$$

where $\vec{n}$ is the unit vector normal to the surface. For a ξ surface,

$$\vec{n} = \frac{\nabla\xi}{|\nabla\xi|} = \frac{1}{m}\xi_x\vec{i} + \frac{1}{m}\xi_y\vec{j} + \frac{1}{m}\xi_z\vec{k}$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}$$

Therefore, for a ξ surface,

$$V_n = \frac{1}{m}(\xi_x u + \xi_y v + \xi_z w)$$

Note that the unit vector $\vec{n}$ is in the direction of increasing ξ. Therefore V_n is positive in the direction of increasing ξ.

Similarly, for an η boundary,

$$V_n = \frac{1}{m}(\eta_x u + \eta_y v + \eta_z w)$$

where

$$m = \sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}$$

and V_n is positive in the direction of increasing η.

And, for a ζ boundary,

$$V_n = \frac{1}{m}(\zeta_x u + \zeta_y v + \zeta_z w)$$

where

$$m = \sqrt{\zeta_x^{\,2} + \zeta_y^{\,2} + \zeta_z^{\,2}}$$

and V_n is positive in the direction of increasing ζ.

Subroutine BCVVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute y-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRV	Array indices associated with the dependent variables ρ, ρu, and ρv.
RHO, U, V	Static density ρ, and velocities u and v, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCVVEL computes coefficients and source terms for y-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular y-velocity boundary conditions in *Proteus*.[13]

[13] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

No Change From Initial Conditions, $\Delta v = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial \hat{Q} = J \partial g/\partial Q$, we get simply

$$J_{i,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{i,j,k} = 0$$

Specified y-Velocity, $v = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{i,j,k} = f^{n+1}_{i,j,k} - v^n_{i,j,k}$$

Specified Two-Point y-Velocity Gradient in Coordinate Direction, $\partial v/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{1,j,k} + J_{2,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{2,j,k} =$$
$$(\Delta \xi) f^{n+1}_{1,j,k} + v^n_{1,j,k} - u^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{N_1-1,j,k} + J_{N_1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{N_1,j,k} =$$
$$(\Delta \xi) f^{n+1}_{N_1,j,k} + v^n_{N_1-1,j,k} - v^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point y-Velocity Gradient in Coordinate Direction, $\partial v/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{1,j,k} + 4J_{2,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{2,j,k}$$
$$-J_{3,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{3,j,k} = 2(\Delta \xi) f^{n+1}_{1,j,k} + 3v^n_{1,j,k} - 4v^n_{2,j,k} + v^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{N_1-1,j,k}$$
$$+ 3J_{N_1,j,k} \left[-\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho v}) \right]^n_{N_1,j,k} = 2(\Delta \xi) f^{n+1}_{N_1,j,k} - v^n_{N_1-2,j,k} + 4v^n_{N_1-1,j,k} - 3v^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point y-Velocity Gradient in Normal Direction, $\nabla v \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- J_{1,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{1,j,k} + J_{2,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{2,j,k} = \\
&\quad \frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta v^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta v^n_{1,j,k} \right] \\
&\qquad + v^n_{1,j,k} - v^n_{2,j,k}
\end{aligned}
$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{N_1-1,j,k} + J_{N_1,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{N_1,j,k} = \\
&\quad \frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta v^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta v^n_{N_1,j,k} \right] \\
&\qquad + v^n_{N_1-1,j,k} - v^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point y-Velocity Gradient in Normal Direction, $\nabla v \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{1,j,k} + 4J_{2,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{2,j,k} \\
&- J_{3,j,k}\left[-\frac{v}{\rho}\,\Delta\hat{\rho} + \frac{1}{\rho}\,\Delta(\hat{\rho v}) \right]^n_{3,j,k} = \\
&\quad \frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta v^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta v^n_{1,j,k} \right] \\
&\qquad + 3v^n_{1,j,k} - 4v^n_{2,j,k} + v^n_{3,j,k}
\end{aligned}
$$

where

$$m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1-1,j,k}$$

$$+ 3J_{N_1,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1,j,k} =$$

$$\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta v^n_{N_1,j,k} - \frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta v^n_{N_1,j,k}\right]$$

$$- v^n_{N_1-2,j,k} + 4v^n_{N_1-1,j,k} - 3v^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of y-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$J_{1,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{1,j,k} - 2J_{2,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{2,j,k}$$

$$+ J_{3,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{3,j,k} = - v^n_{1,j,k} + 2v^n_{2,j,k} - v^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1-2,j,k} - 2J_{N_1-1,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1-1,j,k}$$

$$+ J_{N_1,j,k}\left[-\frac{v}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho v})\right]^n_{N_1,j,k} = - v^n_{N_1-2,j,k} + 2v^n_{N_1-1,j,k} - v^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Flow Angle, $\tan^{-1}(v/u) = f$

This boundary condition can be rewritten as

$$\frac{v}{u} = \tan f$$

where f is the specified flow angle. Multiplying by ρu,

$$(\tan f)\rho u - \rho v = 0$$

Applying equation (6.5) of Volume 1 to the above equation, we get

$$J_{i,j,k}\left[(\tan f)^{n+1}_{i,j,k}\Delta(\hat{\rho u})^n_{i,j,k} - \Delta(\hat{\rho v})^n_{i,j,k}\right] = -(\tan f)^n_{i,j,k} + (\rho v)^n_{i,j,k}$$

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent y-velocity boundary condition is specified.

Subroutine BCV1 (J1,J2,J3,VC1)		
Called by	Calls	Purpose
BC1VEL		Compute ξ-velocity.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
J1, J2, J3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
U, V, W	Velocities u, v, and w, at time level n.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VC1	Velocity in the ξ direction.

Description

Subroutine BCV1 computes the velocity in the ξ direction. The ξ-velocity is defined as

$$V_\xi = \vec{V} \cdot \vec{e}_\xi$$

where $\vec{e}_\xi$ is the unit vector in the ξ direction, given by,

$$\vec{e}_\xi = \frac{1}{m}\left(x_\xi \vec{i} + y_\xi \vec{j} + z_\xi \vec{k}\right)$$

where

$$m = \sqrt{x_\xi^2 + y_\xi^2 + z_\xi^2}$$

Therefore,

$$V_\xi = \frac{1}{m}\left(x_\xi u + y_\xi v + z_\xi w\right)$$

Subroutine BCV2 (J1,J2,J3,VC2)		
Called by	Calls	Purpose
BC2VEL		Compute η-velocity.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
J1, J2, J3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
U, V, W	Velocities u, v, and w, at time level n.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VC2	Velocity in the η direction.

Description

Subroutine BCV2 computes the velocity in the η direction. The η-velocity is defined as

$$V_\eta = \vec{V} \cdot \vec{e}_\eta$$

where $\vec{e}_\eta$ is the unit vector in the η direction, given by,

$$\vec{e}_\eta = \frac{1}{m}\left(x_\eta \vec{i} + y_\eta \vec{j} + z_\eta \vec{k}\right)$$

where

$$m = \sqrt{x_\eta^2 + y_\eta^2 + z_\eta^2}$$

Therefore,

$$V_\eta = \frac{1}{m}\,(x_\eta u + y_\eta v + z_\eta w)$$

Subroutine BCV3 (J1,J2,J3,VC3)		
Called by	Calls	Purpose
BC3VEL		Compute ζ-velocity.

Input

ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
J1, J2, J3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
U, V, W	Velocities u, v, and w, at time level n.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VC3	Velocity in the ζ direction.

Description

Subroutine BCV3 computes the velocity in the ζ direction. The ζ-velocity is defined as

$$V_\zeta = \vec{V} \cdot \vec{e}_\zeta$$

where $\vec{e}_\zeta$ is the unit vector in the ζ direction, given by,

$$\vec{e}_\zeta = \frac{1}{m}\left(x_\zeta \vec{i} + y_\zeta \vec{j} + z_\zeta \vec{k}\right)$$

where

$$m = \sqrt{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}$$

Therefore,

$$V_\zeta = \frac{1}{m}\left(x_\zeta u + y_\zeta v + z_\zeta w\right)$$

Subroutine BCWVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCGRAD BCMET	Compute z-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRW	Array indices associated with the dependent variables ρ, ρu, and ρw.
RHO, U, W	Static density ρ, and velocities u and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BCWVEL computes coefficients and source terms for z-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular z-velocity boundary conditions in *Proteus*.[14]

[14] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

No Change From Initial Conditions, $\Delta w = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial\hat{\mathbf{Q}} = J\partial g/\partial\mathbf{Q}$, we get simply

$$J_{i,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{i,j,k} = 0$$

Specified z-Velocity, $w = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{i,j,k} = f^{n+1}_{i,j,k} - w^n_{i,j,k}$$

Specified Two-Point z-Velocity Gradient in Coordinate Direction, $\partial w/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} + J_{2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} =$$
$$(\Delta\xi)f^{n+1}_{1,j,k} + w^n_{1,j,k} - w^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} + J_{N_1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$(\Delta\xi)f^{n+1}_{N_1,j,k} + w^n_{N_1-1,j,k} - w^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point z-Velocity Gradient in Coordinate Direction, $\partial w/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} + 4J_{2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k}$$
$$-J_{3,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} = 2(\Delta\xi)f^{n+1}_{1,j,k} + 3w^n_{1,j,k} - 4w^n_{2,j,k} + w^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}$$
$$+ 3J_{N_1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} = 2(\Delta\xi)f^{n+1}_{N_1,j,k} - w^n_{N_1-2,j,k} + 4w^n_{N_1-1,j,k} - 3w^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point z-Velocity Gradient in Normal Direction, $\nabla w \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- J_{1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{1,j,k} + J_{2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta w^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta w^n_{1,j,k}\right] \\
&\qquad + w^n_{1,j,k} - w^n_{2,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
&- J_{N_1-1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{N_1-1,j,k} + J_{N_1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{N_1,j,k} = \\
&\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta w^n_{N_1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta w^n_{N_1,j,k}\right] \\
&\qquad + w^n_{N_1-1,j,k} - w^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point z-Velocity Gradient in Normal Direction, $\nabla w \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3J_{1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{1,j,k} + 4J_{2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{2,j,k} \\
&- J_{3,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho} + \frac{1}{\rho}\Delta(\widehat{\rho w})\right]^n_{3,j,k} = \\
&\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta w^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta w^n_{1,j,k}\right] \\
&\qquad + 3w^n_{1,j,k} - 4w^n_{2,j,k} + w^n_{3,j,k}
\end{aligned}
$$

where

$$
m = \sqrt{\xi_x^{\,2} + \xi_y^{\,2} + \xi_z^{\,2}}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} - 4J_{N_1-1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}$$

$$+\,3J_{N_1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$

$$\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k} - \frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta w^n_{N_1,j,k} - \frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta w^n_{N_1,j,k}\right]$$

$$- w^n_{N_1-2,j,k} + 4w^n_{N_1-1,j,k} - 3w^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of z-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$J_{1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} - 2J_{2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k}$$

$$+\,J_{3,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} = -w^n_{1,j,k} + 2w^n_{2,j,k} - w^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} - 2J_{N_1-1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}$$

$$+\,J_{N_1,j,k}\left[-\frac{w}{\rho}\Delta\hat{\rho}+\frac{1}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} = -w^n_{N_1-2,j,k} + 2w^n_{N_1-1,j,k} - w^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Flow Angle, $\tan^{-1}(w/u) = f$

This boundary condition can be rewritten as

$$\frac{w}{u} = \tan f$$

where f is the specified flow angle. Multiplying by ρu,

$$(\tan f)\rho u - \rho w = 0$$

Applying equation (6.5) of Volume 1 to the above equation, we get

$$J_{i,j,k}\left[(\tan f)^{n+1}_{i,j,k}\Delta(\hat{\rho u})^n_{i,j,k} - \Delta(\hat{\rho w})^n_{i,j,k}\right] = -(\tan f)^n_{i,j,k} + (\rho w)^n_{i,j,k}$$

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent z-velocity boundary condition is specified.

Subroutine BC1VEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCIMET BCMET BCV1	Compute ξ-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW	Array indices associated with the dependent variables ρ, ρu, ρv, and ρw.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BC1VEL computes coefficients and source terms for ξ-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular ξ-direction velocity boundary conditions in *Proteus*.[15]

[15] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript * and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the

Specified ξ-Velocity, $V_\xi = f$

The velocity in the ξ direction, V_ξ, is defined as

$$V_\xi = \vec{V} \cdot \vec{e}_\xi$$

where $\vec{e}_\xi$ is the unit vector in the ξ direction, given by,

$$\vec{e}_\xi = \frac{1}{m}\left(x_\xi \vec{i} + y_\xi \vec{j} + z_\xi \vec{k}\right)$$

where

$$m = \sqrt{x_\xi^2 + y_\xi^2 + z_\xi^2}$$

Therefore,

$$V_\xi = \frac{1}{m}\left(x_\xi u + y_\xi v + z_\xi w\right)$$

Applying equation (6.5) of Volume 1, the linearized boundary condition at a ξ boundary becomes

$$\frac{J_{i,j,k}}{m_{i,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\widehat{\rho w})\right]_{i,j,k}^n = f_{i,j,k}^{n+1} - (V_\xi)_{i,j,k}^n$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point ξ-Velocity Gradient in Coordinate Direction, $\partial V_\xi / \partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\begin{aligned}
&-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\widehat{\rho w})\right]_{1,j,k}^n + \\
&\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\widehat{\rho w})\right]_{2,j,k}^n = \\
&\qquad (\Delta\xi) f_{1,j,k}^{n+1} + (V_\xi)_{1,j,k}^n - (V_\xi)_{2,j,k}^n
\end{aligned}$$

At the $\xi = 1$ boundary,

$$\begin{aligned}
&-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\widehat{\rho w})\right]_{N_1-1,j}^n + \\
&\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\widehat{\rho w})\right]_{N_1,j,k}^n = \\
&\qquad (\Delta\xi) f_{N_1,j,k}^{n+1} + (V_\xi)_{N_1-1,j,k}^n - (V_\xi)_{N_1,j,k}^n
\end{aligned}$$

Analogous equations can easily be written for the η and ζ boundaries.

superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

Specified Three-Point ξ-Velocity Gradient in Coordinate Direction, $\partial V_\xi / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&- 3 \frac{J_{1,j,k}}{m_{1,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{1,j,k} + \\
&4 \frac{J_{2,j,k}}{m_{2,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{2,j,k} - \\
&\frac{J_{3,j,k}}{m_{3,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{3,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{1,j,k} + 3(V_\xi)^n_{1,j,k} - 4(V_\xi)^n_{2,j,k} + (V_\xi)^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1-2,j,k} - \\
&4 \frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1-1,j,k} + \\
&3 \frac{J_{N_1,j,k}}{m_{N_1,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1,j,k} = \\
&\qquad 2(\Delta\xi) f^{n+1}_{N_1,j,k} - (V_\xi)^n_{N_1-2,j,k} + 4(V_\xi)^n_{N_1-1,j,k} - 3(V_\xi)^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point ξ-Velocity Gradient in Normal Direction, $\nabla V_\xi \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&- \frac{J_{1,j,k}}{m_{1,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{1,j,k} + \\
&\frac{J_{2,j,k}}{m_{2,j,k}} \left[- \frac{x_\xi u + y_\xi v + z_\xi w}{\rho} \Delta\hat{\rho} + \frac{x_\xi}{\rho} \Delta(\hat{\rho u}) + \frac{y_\xi}{\rho} \Delta(\hat{\rho v}) + \frac{z_\xi}{\rho} \Delta(\hat{\rho w}) \right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}} \left[f^{n+1}_{1,j,k} - \frac{(\xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z)_{1,j,k}}{m_{1,j,k}} \delta_\eta (V_\xi)^n_{1,j,k} - \frac{(\xi_x \zeta_x + \xi_y \zeta_y + \xi_z \zeta_z)_{1,j,k}}{m_{1,j,k}} \delta_\zeta (V_\xi)^n_{1,j,k} \right] \\
&\qquad + (V_\xi)^n_{1,j,k} - (V_\xi)^n_{2,j,k}
\end{aligned}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}+
$$

$$
\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k}=
$$

$$
\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta(V_\xi)^n_{N_1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta(V_\xi)^n_{N_1,j,k}\right]
$$

$$
+(V_\xi)^n_{N_1-1,j,k}-(V_\xi)^n_{N_1,j,k}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point ξ-Velocity Gradient in Normal Direction, $\nabla V_\xi \cdot \vec{n}=f$

Applying equation (6.12a) of Volume 1 at the $\xi=0$ boundary, and using three-point one-sided differencing,

$$
-3\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k}+
$$

$$
4\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k}-
$$

$$
\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k}=
$$

$$
\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta(V_\xi)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta(V_\xi)^n_{1,j,k}\right]
$$

$$
+3(V_\xi)^n_{1,j,k}-4(V_\xi)^n_{2,j,k}+(V_\xi)^n_{3,j,k}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi=1$ boundary,

$$
\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k}-
$$

$$
4\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}+
$$

$$
3\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\xi u+y_\xi v+z_\xi w}{\rho}\Delta\hat{\rho}+\frac{x_\xi}{\rho}\Delta(\hat{\rho u})+\frac{y_\xi}{\rho}\Delta(\hat{\rho v})+\frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k}=
$$

$$
\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\eta(V_\xi)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\delta_\zeta(V_\xi)^n_{N_1,j,k}\right]
$$

$$
-(V_\xi)^n_{N_1-2,j,k}+4(V_\xi)^n_{N_1-1,j,k}-3(V_\xi)^n_{N_1,j,k}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of ξ-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} -$$
$$2\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} +$$
$$\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} =$$
$$-(V_\xi)^n_{1,j,k} + 2(V_\xi)^n_{2,j,k} - (V_\xi)^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} -$$
$$2\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} +$$
$$\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\xi u + y_\xi v + z_\xi w}{\rho}\Delta\hat{\rho} + \frac{x_\xi}{\rho}\Delta(\hat{\rho u}) + \frac{y_\xi}{\rho}\Delta(\hat{\rho v}) + \frac{z_\xi}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$-(V_\xi)^n_{N_1-2,j,k} + 2(V_\xi)^n_{N_1-1,j,k} - (V_\xi)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutin. uses one-dimensional addressing of three-dimensional arrays, as described in S.ction 2.3.
2. An error message is generated and execution is stopped if a non-existent ξ-velocity boundary condition is specified.

Subroutine BC2VEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCIMET BCMET BCV2	Compute η-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
METX, METY, METZ	Derivatives of sweep direction computational coordinate with respect to x, y, and z.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW	Array indices associated with the dependent variables ρ, ρu, ρv, and ρw.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BC2VEL computes coefficients and source terms for η-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular η-direction velocity boundary conditions in *Proteus*.[16]

[16] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript

Specified η-Velocity, $V_\eta = f$

The velocity in the η direction, V_η, is defined as

$$V_\eta = \vec{V} \cdot \vec{e}_\eta$$

where $\vec{e}_\eta$ is the unit vector in the η direction, given by,

$$\vec{e}_\eta = \frac{1}{m}\left(x_\eta \vec{i} + y_\eta \vec{j} + z_\eta \vec{k}\right)$$

where

$$m = \sqrt{x_\eta^2 + y_\eta^2 + z_\eta^2}$$

Therefore,

$$V_\eta = \frac{1}{m}(x_\eta u + y_\eta v + z_\eta w)$$

Applying equation (6.5) of Volume 1, the linearized boundary condition at a ξ boundary becomes

$$\frac{J_{i,j,k}}{m_{i,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{i,j,k} = f^{n+1}_{i,j,k} - (V_\eta)^n_{i,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point η-Velocity Gradient in Coordinate Direction, $\partial V_\eta / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} +$$
$$\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} =$$
$$(\Delta\xi) f^{n+1}_{1,j,k} + (V_\eta)^n_{1,j,k} - (V_\eta)^n_{2,j,k}$$

At the $\xi = 1$ boundary,

$$-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j} +$$
$$\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$(\Delta\xi) f^{n+1}_{N_1,j,k} + (V_\eta)^n_{N_1-1,j,k} - (V_\eta)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

---

* and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

Specified Three-Point η-Velocity Gradient in Coordinate Direction, $\partial V_\eta / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- 3 \frac{J_{1,j,k}}{m_{1,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{1,j,k} +$$

$$4 \frac{J_{2,j,k}}{m_{2,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{2,j,k} -$$

$$\frac{J_{3,j,k}}{m_{3,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{3,j,k} =$$

$$2(\Delta\xi) f^{n+1}_{1,j,k} + 3(V_\eta)^n_{1,j,k} - 4(V_\eta)^n_{2,j,k} + (V_\eta)^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1-2,j,k} -$$

$$4 \frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1-1,j,k} +$$

$$3 \frac{J_{N_1,j,k}}{m_{N_1,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{N_1,j,k} =$$

$$2(\Delta\xi) f^{n+1}_{N_1,j,k} - (V_\eta)^n_{N_1-2,j,k} + 4(V_\eta)^n_{N_1-1,j,k} - 3(V_\eta)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point η-Velocity Gradient in Normal Direction, $\nabla V_\eta \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$- \frac{J_{1,j,k}}{m_{1,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{1,j,k} +$$

$$\frac{J_{2,j,k}}{m_{2,j,k}} \left[- \frac{x_\eta u + y_\eta v + z_\eta w}{\rho} \Delta\hat{\rho} + \frac{x_\eta}{\rho} \Delta(\hat{\rho u}) + \frac{y_\eta}{\rho} \Delta(\hat{\rho v}) + \frac{z_\eta}{\rho} \Delta(\hat{\rho w}) \right]^n_{2,j,k} =$$

$$\frac{\Delta\xi}{m_{1,j,k}} \left[f^{n+1}_{1,j,k} - \frac{(\xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z)_{1,j,k}}{m_{1,j,k}} \delta_\eta (V_\eta)^n_{1,j,k} - \frac{(\xi_x \zeta_x + \xi_y \zeta_y + \xi_z \zeta_z)_{1,j,k}}{m_{1,j,k}} \delta_\zeta (V_\eta)^n_{1,j,k} \right]$$
$$+ (V_\eta)^n_{1,j,k} - (V_\eta)^n_{2,j,k}$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}+
$$

$$
\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{N_1,j,k}=
$$

$$
\frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta(V_\eta)^n_{N_1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta(V_\eta)^n_{N_1,j,k}\right]
$$

$$
+(V_\eta)^n_{N_1-1,j,k}-(V_\eta)^n_{N_1,j,k}
$$

Analogous equations can easily be written for the η and ζ boundaries.

<u>*Specified Three-Point η-Velocity Gradient in Normal Direction, $\nabla V_\eta\cdot\vec{n}=f$*</u>

Applying equation (6.12a) of Volume 1 at the $\xi=0$ boundary, and using three-point one-sided differencing,

$$
-3\,\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{1,j,k}+
$$

$$
4\,\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{2,j,k}-
$$

$$
\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{3,j,k}=
$$

$$
\frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta(V_\eta)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta(V_\eta)^n_{1,j,k}\right]
$$

$$
+3(V_\eta)^n_{1,j,k}-4(V_\eta)^n_{2,j,k}+(V_\eta)^n_{3,j,k}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi=1$ boundary,

$$
\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k}-
$$

$$
4\,\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k}+
$$

$$
3\,\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\eta u+y_\eta v+z_\eta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\eta}{\rho}\,\Delta(\hat{\rho u})+\frac{y_\eta}{\rho}\,\Delta(\hat{\rho v})+\frac{z_\eta}{\rho}\,\Delta(\hat{\rho w})\right]^n_{N_1,j,k}=
$$

$$
\frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta(V_\eta)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta(V_\eta)^n_{N_1,j,k}\right]
$$

$$
-(V_\eta)^n_{N_1-2,j,k}+4(V_\eta)^n_{N_1-1,j,k}-3(V_\eta)^n_{N_1,j,k}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of η-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} -$$

$$2\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} +$$

$$\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} =$$

$$-(V_\eta)^n_{1,j,k} + 2(V_\eta)^n_{2,j,k} - (V_\eta)^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} -$$

$$2\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} +$$

$$\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\eta u + y_\eta v + z_\eta w}{\rho}\Delta\hat{\rho} + \frac{x_\eta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\eta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\eta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$

$$-(V_\eta)^n_{N_1-2,j,k} + 2(V_\eta)^n_{N_1-1,j,k} - (V_\eta)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent η-velocity boundary condition is specified.

Subroutine BC3VEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND)		
Called by	Calls	Purpose
BCGEN	BCIMET BCMET BCV3	Compute ζ-velocity boundary conditions.

Input

DEL	Computational grid spacing in sweep direction.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
IBC, FBC	Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to N_{eq}, corresponding to the N_{eq} conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
IBOUND	Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ	Boundary condition equation number.
IMIN, IMAX	Minimum and maximum indices in the sweep direction.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
METX, METY, METZ	Derivatives of sweep direction computational coordinate with respect to x, y, and z.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW	Array indices associated with the dependent variables ρ, ρu, ρv, and ρw.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at boundary IBOUND (row IEQ only).
S	Source term subvector **S** at boundary IBOUND (element IEQ only).

Description

Subroutine BC3VEL computes coefficients and source terms for ζ-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular ζ-direction velocity boundary conditions in *Proteus*.[17]

[17] In the following description, for the first and second ADI sweeps the dependent variable should have the superscript

Specified ζ-Velocity, $V_\zeta = f$

The velocity in the ζ direction, V_ζ, is defined as

$$V_\zeta = \vec{V} \cdot \vec{e}_\zeta$$

where $\vec{e}_\zeta$ is the unit vector in the ζ direction, given by,

$$\vec{e}_\zeta = \frac{1}{m}\left(x_\zeta \vec{i} + y_\zeta \vec{j} + z_\zeta \vec{k}\right)$$

where

$$m = \sqrt{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}$$

Therefore,

$$V_\zeta = \frac{1}{m}(x_\zeta u + y_\zeta v + z_\zeta w)$$

Applying equation (6.5) of Volume 1, the linearized boundary condition at a ξ boundary becomes

$$\frac{J_{i,j,k}}{m_{i,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\widehat{\rho w})\right]_{i,j,k}^n = f_{i,j,k}^{n+1} - (V_\zeta)_{i,j,k}^n$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point ζ-Velocity Gradient in Coordinate Direction, $\partial V_\zeta / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\widehat{\rho w})\right]_{1,j,k}^n +$$

$$\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\widehat{\rho w})\right]_{2,j,k}^n =$$

$$(\Delta\xi) f_{1,j,k}^{n+1} + (V_\zeta)_{1,j,k}^n - (V_\zeta)_{2,j,k}^n$$

At the $\xi = 1$ boundary,

$$-\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\widehat{\rho w})\right]_{N_1-1,j}^n +$$

$$\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\widehat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\widehat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\widehat{\rho w})\right]_{N_1,j,k}^n =$$

$$(\Delta\xi) f_{N_1,j,k}^{n+1} + (V_\zeta)_{N_1-1,j,k}^n - (V_\zeta)_{N_1,j,k}^n$$

Analogous equations can easily be written for the η and ζ boundaries.

---

* and **, respectively, representing the intermediate solution, and for the third ADI sweep it should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.

Specified Three-Point ζ-Velocity Gradient in Coordinate Direction, $\partial V_\zeta/\partial\phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
&-3\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} + \\
&4\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} - \\
&\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} = \\
&\qquad 2(\Delta\xi)f^{n+1}_{1,j,k} + 3(V_\zeta)^n_{1,j,k} - 4(V_\zeta)^n_{2,j,k} + (V_\zeta)^n_{3,j,k}
\end{aligned}
$$

At the $\xi = 1$ boundary,

$$
\begin{aligned}
&\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} - \\
&4\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} + \\
&3\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} = \\
&\qquad 2(\Delta\xi)f^{n+1}_{N_1,j,k} - (V_\zeta)^n_{N_1-2,j,k} + 4(V_\zeta)^n_{N_1-1,j,k} - 3(V_\zeta)^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Two-Point ζ-Velocity Gradient in Normal Direction, $\nabla V_\zeta \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$
\begin{aligned}
&-\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} + \\
&\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} = \\
&\frac{\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k} - \frac{(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\delta_\eta(V_\zeta)^n_{1,j,k} - \frac{(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\delta_\zeta(V_\zeta)^n_{1,j,k}\right] \\
&\qquad + (V_\zeta)^n_{1,j,k} - (V_\zeta)^n_{2,j,k}
\end{aligned}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$
\begin{aligned}
& -\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{N_1-1,j,k}+ \\
& \frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{N_1,j,k}= \\
& \frac{\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta(V_\zeta)^n_{N_1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta(V_\zeta)^n_{N_1,j,k}\right] \\
& \quad +(V_\zeta)^n_{N_1-1,j,k}-(V_\zeta)^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Specified Three-Point ζ-Velocity Gradient in Normal Direction, $\nabla V_\zeta \cdot \vec{n}=f$

Applying equation (6.12a) of Volume 1 at the $\xi=0$ boundary, and using three-point one-sided differencing,

$$
\begin{aligned}
& -3\,\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{1,j,k}+ \\
& 4\,\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{2,j,k}- \\
& \frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{3,j,k}= \\
& \frac{2\Delta\xi}{m_{1,j,k}}\left[f^{n+1}_{1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\eta(V_\zeta)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{1,j,k}}{m_{1,j,k}}\,\delta_\zeta(V_\zeta)^n_{1,j,k}\right] \\
& \quad +3(V_\zeta)^n_{1,j,k}-4(V_\zeta)^n_{2,j,k}+(V_\zeta)^n_{3,j,k}
\end{aligned}
$$

and δ_η and δ_ζ are the centered difference operators presented in Section 5.0 of Volume 1. At the $\xi=1$ boundary,

$$
\begin{aligned}
& \frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{N_1-2,j,k}- \\
& 4\,\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{N_1-1,j,k}+ \\
& 3\,\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\zeta u+y_\zeta v+z_\zeta w}{\rho}\,\Delta\hat{\rho}+\frac{x_\zeta}{\rho}\,\Delta(\widehat{\rho u})+\frac{y_\zeta}{\rho}\,\Delta(\widehat{\rho v})+\frac{z_\zeta}{\rho}\,\Delta(\widehat{\rho w})\right]^n_{N_1,j,k}= \\
& \frac{2\Delta\xi}{m_{N_1,j,k}}\left[f^{n+1}_{N_1,j,k}-\frac{(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\eta(V_\zeta)^n_{1,j,k}-\frac{(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z)_{N_1,j,k}}{m_{N_1,j,k}}\,\delta_\zeta(V_\zeta)^n_{N_1,j,k}\right] \\
& \quad -(V_\zeta)^n_{N_1-2,j,k}+4(V_\zeta)^n_{N_1-1,j,k}-3(V_\zeta)^n_{N_1,j,k}
\end{aligned}
$$

Analogous equations can easily be written for the η and ζ boundaries.

Linear Extrapolation of ζ-Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$\frac{J_{1,j,k}}{m_{1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{1,j,k} -$$
$$2\frac{J_{2,j,k}}{m_{2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{2,j,k} +$$
$$\frac{J_{3,j,k}}{m_{3,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{3,j,k} =$$
$$-(V_\zeta)^n_{1,j,k} + 2(V_\zeta)^n_{2,j,k} - (V_\zeta)^n_{3,j,k}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j,k}}{m_{N_1-2,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-2,j,k} -$$
$$2\frac{J_{N_1-1,j,k}}{m_{N_1-1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1-1,j,k} +$$
$$\frac{J_{N_1,j,k}}{m_{N_1,j,k}}\left[-\frac{x_\zeta u + y_\zeta v + z_\zeta w}{\rho}\Delta\hat{\rho} + \frac{x_\zeta}{\rho}\Delta(\hat{\rho u}) + \frac{y_\zeta}{\rho}\Delta(\hat{\rho v}) + \frac{z_\zeta}{\rho}\Delta(\hat{\rho w})\right]^n_{N_1,j,k} =$$
$$-(V_\zeta)^n_{N_1-2,j,k} + 2(V_\zeta)^n_{N_1-1,j,k} - (V_\zeta)^n_{N_1,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent ζ-velocity boundary condition is specified.

Subroutine BLIN		
Called by	Calls	Purpose
TURBBL	ISRCHFGT VORTEX	Compute inner layer turbulent viscosity.

Input

* APLUS	Van Driest damping constant A^+.
* CB	Constant B in the Spalding-Kleinstein inner layer model.
* CNL	Exponent n in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model.
* CVK	Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein models.
EP1, EP2	Minimum and maximum allowable numerical values.
* ILDAMP	Flag for Launder-Priddin modified mixing length formula in the Baldwin-Lomax inner region model.
* INNER	Flag for type of inner region model.
* LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries.
MU	Laminar coefficient of viscosity μ_l.
MUT	Outer layer turbulent viscosity coefficient $(\mu_t)_{outer}$.
NTOTP	Dimensioning parameter specifying the storage required for a full three-dimensional array (i.e., N1P $\times$ N2P $\times$ N3P).
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
* RER	Reference Reynolds number Re_r.
RHO, U, V, W	Static density ρ, and velocities u, v, and w.
X, Y, Z	Cartesian coordinates x, y, and z.

Output

MUT	Turbulent viscosity coefficient μ_t.

Description

Subroutine BLIN computes the inner layer turbulent viscosity coefficient $(\mu_t)_{inner}$. For each grid point, subroutine BLIN first sets the variable DUMMY equal to a number from 1.0 to 6.0 as a flag specifying the nearest solid wall. If none of the three grid lines through the point intersect a solid wall, the point is a wake point and DUMMY = − 1. If there are no solid walls, control is returned to the calling program. Otherwise, subroutine VORTEX is called to get the vorticity magnitude at each point.

The inner layer turbulent viscosity coefficient $(\mu_t)_{inner}$ is then computed based on the nearest wall, and it is assumed that the inner regions do not overlap. Three different inner region models are available - the model of Baldwin and Lomax (1978), with and without the modified mixing length formula of Launder and Priddin, and the model of Spalding (1961) and Kleinstein (1967). These are described in Section 9.1 of Volume 1.

BLIN then sets the final turbulent viscosity coefficient equal to the minimum of the inner and outer region values. Thus,

$$\mu_t = \min[(\mu_t)_{inner}, (\mu_t)_{outer}]$$

Remarks

1. To avoid the possibility of floating point errors, the value of $\left|\vec{\Omega}\right|_w$ used to compute τ^+ and u^+ is set to a minimum of 10^{-10}.

Subroutine BLKOUT (I1PT,I2PT,I3PT)		
Called by	**Calls**	**Purpose**
ADI AVISC1 AVISC2 BCGEN FILTER		Print coefficient blocks at specified indices in the ξ, η, and ζ directions.

Input

	A, B, C	Coefficient submatrices **A**, **B**, and **C**
*	IHSTAG	Flag for constant stagnation enthalpy option.
	ISWEEP	Current ADI sweep number.
	I1PT, I2PT, I3PT	Indices for printout in the ξ, η, and ζ directions.
	NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations.
	NEQ	Number of coupled equations being solved, N_{eq}.
*	NOUT	Unit number for standard output.
	S	Source term subvector **S**.

Output

None.

Description

Subroutine BLKOUT prints the coefficient block submatrices **A**, **B**, and **C**, and the source term subvector **S** at the grid points specified by I1PT, I2PT, and I3PT. This is the routine that actually prints the output for the IDEBUG(1) through IDEBUG(4) options.

Subroutine BLK4		
Called by	Calls	Purpose
ADI	FILTER	Solve 4 × 4 block tridiagonal system of equations.

Input

A, B, C	Coefficient submatrices **A**, **B**, and **C**
NPTS	Number of grid points in the sweep direction, N.
NV	Number of grid points in the "vectorized" direction, N_v.
S	Source term subvector **S**.

Output

S	Computed solution subvector.

Description

Subroutine BLK4 solves a block tridiagonal system of equations with 4 × 4 blocks using the block matrix version of the Thomas algorithm. Subroutine FILTER is called in an attempt to eliminate any zero values on the diagonal of the submatrix **B** at the two boundaries. These can occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices **D**, **E**, and $\Delta\hat{\mathbf{Q}}'$. In Fortran, however, we can save storage by overwriting B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second and third sweeps.

Step	In Fortran	In Volume 1 Notation
1		$\mathbf{D}_1 = \mathbf{B}_1$
2a	LU decompose $\mathbf{B}_1$, storing result in $\mathbf{B}_1$	LU decomposition of $\mathbf{D}_1$
2b	Solve $\mathbf{B}_1\mathbf{E}_1 = \mathbf{C}_1$ for $\mathbf{E}_1$ using LU decomposition of $\mathbf{B}_1$, storing result in $\mathbf{C}_1$	$\mathbf{E}_1 = \mathbf{D}_1^{-1}\mathbf{C}_1$
2c	Solve $\mathbf{B}_1\Delta\hat{\mathbf{Q}}_1' = \mathbf{S}_1$ for $\Delta\hat{\mathbf{Q}}_1'$ using LU decomposition of $\mathbf{B}_1$, storing result in $\mathbf{S}_1$	$\Delta\hat{\mathbf{Q}}_1' = \mathbf{D}_1^{-1}\mathbf{S}_1$
	For $i = 2$ to N_1,	
3a	Compute $\mathbf{B}_i - \mathbf{A}_i\mathbf{C}_{i-1}$, storing result in $\mathbf{B}_i$	$\mathbf{D}_i = \mathbf{B}_i - \mathbf{A}_i\mathbf{E}_{i-1}$
3b	Compute $\mathbf{S}_i - \mathbf{A}_i\mathbf{S}_{i-1}$, storing result in $\mathbf{S}_i$	$\mathbf{S}_i - \mathbf{A}_i\Delta\hat{\mathbf{Q}}_{i-1}'$
3c	LU decompose $\mathbf{B}_i$, storing result in $\mathbf{B}_i$	LU decomposition of $\mathbf{D}_i$
3d	Solve $\mathbf{B}_i\mathbf{E}_i = \mathbf{C}_i$ for $\mathbf{E}_i$ using LU decomposition of $\mathbf{B}_i$, storing result in $\mathbf{C}_i$	$\mathbf{E}_i = \mathbf{D}_i^{-1}\mathbf{C}_i$
3e	Solve $\mathbf{B}_i\Delta\hat{\mathbf{Q}}_i' = \mathbf{S}_i$ for $\Delta\hat{\mathbf{Q}}_i'$ using LU decomposition of $\mathbf{B}_i$, storing result in $\mathbf{S}_i$	$\Delta\hat{\mathbf{Q}}_i' = \mathbf{D}_i^{-1}(\mathbf{S}_i - \mathbf{A}_i\Delta\hat{\mathbf{Q}}_{i-1}')$
4		$\Delta\hat{\mathbf{Q}}_{N_1} = \Delta\hat{\mathbf{Q}}_{N_1}'$
	For $i = N_1 - 1$ to 1,	
5	Compute $\mathbf{S}_i - \mathbf{C}_i\mathbf{S}_{i+1}$, storing result in $\mathbf{S}_i$	$\Delta\hat{\mathbf{Q}}_i = \Delta\hat{\mathbf{Q}}_i' - \mathbf{E}_i\Delta\hat{\mathbf{Q}}_{i+1}$

Remarks

1. The notation used in the comments in BLK4 is consistent with the notation used in the description of the algorithm in Volume 1.
2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in all three directions, the algorithm can be vectorized in one of the non-sweep directions. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

Subroutine BLK4P		
Called by	Calls	Purpose
ADI		Solve 4×4 periodic block tridiagonal system of equations.

Input

A, B, C	Coefficient submatrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$
NPTS	Number of grid points in the sweep direction, N.
NV	Number of grid points in the "vectorized" direction, N_v.
S	Source term subvector $\mathbf{S}$.

Output

S	Computed solution subvector.

Description

Subroutine BLK4P solves a periodic block tridiagonal system of equations with 4×4 blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices $\mathbf{D}$, $\mathbf{E}$, $\mathbf{F}$, $\mathbf{G}$, and $\Delta\hat{\mathbf{Q}}'$. In Fortran, however, we can save storage by overwriting A, B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second and third sweeps.

Step	In Fortran	In Volume 1 Notation
1a		$\mathbf{D}_2 = \mathbf{B}_2$
1b		$\mathbf{F}_2 = \mathbf{C}_{N_1}$
2a	LU decompose $\mathbf{B}_2$, storing result in $\mathbf{B}_2$	LU decomposition of $\mathbf{D}_2$
2b	Solve $\mathbf{B}_2\mathbf{E}_2 = \mathbf{C}_2$ for $\mathbf{E}_2$ using LU decomposition of $\mathbf{B}_2$, storing result in $\mathbf{C}_2$	$\mathbf{E}_2 = \mathbf{D}_2^{-1}\mathbf{C}_2$
2c	Solve $\mathbf{B}_2\mathbf{G}_2 = \mathbf{A}_2$ for $\mathbf{G}_2$ using LU decomposition of $\mathbf{B}_2$, storing result in $\mathbf{A}_2$	$\mathbf{G}_2 = \mathbf{D}_2^{-1}\mathbf{A}_2$
2d	Solve $\mathbf{B}_2\Delta\hat{\mathbf{Q}}_2' = \mathbf{S}_2$ for $\Delta\hat{\mathbf{Q}}_2'$ using LU decomposition of $\mathbf{B}_2$, storing result in $\mathbf{S}_2$	$\Delta\hat{\mathbf{Q}}_2' = \mathbf{D}_2^{-1}\mathbf{S}_2$

Step	In Fortran	In Volume 1 Notation
	For $i = 3$ to $N_1 - 1$,	
3a	Compute $\mathbf{B}_i - \mathbf{A}_i\mathbf{C}_{i-1}$, storing result in $\mathbf{B}_i$	$\mathbf{D}_i = \mathbf{B}_i - \mathbf{A}_i\mathbf{E}_{i-1}$
3b	Compute $\mathbf{S}_i - \mathbf{A}_i\mathbf{S}_{i-1}$, storing result in $\mathbf{S}_i$	$\mathbf{S}_i - \mathbf{A}_i\Delta\hat{\mathbf{Q}}'_{i-1}$
3c	Compute $-\mathbf{A}_i\mathbf{A}_{i-1}$, storing result in $\mathbf{A}_i$	$-\mathbf{A}_i\mathbf{G}_{i-1}$
3d	LU decompose $\mathbf{B}_i$, storing result in $\mathbf{B}_i$	LU decomposition of $\mathbf{D}_i$
3e	Solve $\mathbf{B}_i\mathbf{E}_i = \mathbf{C}_i$ for $\mathbf{E}_i$ using LU decomposition of $\mathbf{B}_i$, storing result in $\mathbf{C}_i$	$\mathbf{E}_i = \mathbf{D}_i^{-1}\mathbf{C}_i$
3f	Solve $\mathbf{B}_i\mathbf{G}_i = \mathbf{A}_i$ for $\mathbf{G}_i$ using LU decomposition of $\mathbf{B}_i$, storing result in $\mathbf{A}_i$	$\mathbf{G}_i = \mathbf{D}_i^{-1}\mathbf{A}_i\mathbf{G}_{i-1}$
3g	Solve $\mathbf{B}_i\Delta\hat{\mathbf{Q}}'_i = \mathbf{S}_i$ for $\Delta\hat{\mathbf{Q}}'_i$ using LU decomposition of $\mathbf{B}_i$, storing result in $\mathbf{S}_i$	$\Delta\hat{\mathbf{Q}}'_i = \mathbf{D}_i^{-1}(\mathbf{S}_i - \mathbf{A}_i\Delta\hat{\mathbf{Q}}'_{i-1})$
3h	Compute $\mathbf{B}_{N_1} - \mathbf{C}_{N_1}\mathbf{A}_{i-1}$, storing result in $\mathbf{B}_{N_1}$	$\mathbf{B}_{N_1} - \sum_{j=2}^{i-1}\mathbf{F}_j\mathbf{G}_j$
3i	Compute $\mathbf{S}_{N_1} - \mathbf{C}_{N_1}\mathbf{S}_{i-1}$, storing result in $\mathbf{S}_{N_1}$	$\mathbf{S}_{N_1} - \sum_{j=2}^{i-1}\mathbf{F}_j\Delta\hat{\mathbf{Q}}'_j$
3j	Compute $-\mathbf{C}_{N_1}\mathbf{C}_{i-1}$, storing result in $\mathbf{C}_{N_1}$	$\mathbf{F}_i = -\mathbf{F}_{i-1}\mathbf{E}_{i-1}$
4a	Compute $\mathbf{A}_{N_1-1} + \mathbf{C}_{N_1-1}$, storing result in $\mathbf{A}_{N_1-1}$	$\mathbf{G}_{N_1-1} = \mathbf{D}_{N_1-1}^{-1}(\mathbf{C}_{N_1-1} - \mathbf{A}_{N_1-1}\mathbf{G}_{N_1-2})$
4b	Compute $\mathbf{A}_{N_1} + \mathbf{C}_{N_1}$, storing result in $\mathbf{C}_{N_1}$	$\mathbf{F}_{N_1-1} = \mathbf{A}_{N_1} - \mathbf{F}_{N_1-2}\mathbf{E}_{N_1-2}$
4c	Compute $\mathbf{B}_{N_1} - \mathbf{C}_{N_1}\mathbf{A}_{N_1-1}$, storing result in $\mathbf{B}_{N_1}$	$\mathbf{D}_{N_1} = \mathbf{B}_{N_1} - \sum_{i=2}^{N_1-1}\mathbf{F}_i\mathbf{G}_i$
4d	Compute $\mathbf{S}_{N_1} - \mathbf{C}_{N_1}\mathbf{S}_{N_1-1}$, storing result in $\mathbf{S}_{N_1}$	$\mathbf{S}_{N_1} - \sum_{i=2}^{N_1-1}\mathbf{F}_i\Delta\hat{\mathbf{Q}}'_i$
4e	LU decompose $\mathbf{B}_{N_1}$, storing result in $\mathbf{B}_{N_1}$	LU decomposition of $\mathbf{D}_{N_1}$
4f	Solve $\mathbf{B}_{N_1}\Delta\hat{\mathbf{Q}}'_{N_1} = \mathbf{S}_{N_1}$ for $\Delta\hat{\mathbf{Q}}'_{N_1}$ using LU decomposition of $\mathbf{B}_{N_1}$, storing result in $\mathbf{S}_{N_1}$	$\Delta\hat{\mathbf{Q}}'_{N_1} = \mathbf{D}_{N_1}^{-1}(\mathbf{S}_{N_1} - \sum_{i=2}^{N_1-1}\mathbf{F}_i\Delta\hat{\mathbf{Q}}'_i)$
5		$\Delta\hat{\mathbf{Q}}_{N_1} = \Delta\hat{\mathbf{Q}}'_{N_1}$
6	Compute $\mathbf{S}_{N_1-1} - \mathbf{A}_{N_1-1}\mathbf{S}_{N_1}$, storing result in $\mathbf{S}_{N_1-1}$	$\Delta\hat{\mathbf{Q}}_{N_1-1} = \Delta\hat{\mathbf{Q}}'_{N_1-1} - \mathbf{G}_{N_1-1}\Delta\hat{\mathbf{Q}}_{N_1}$
	For $i = N_1 - 2$ to 2,	
7	Compute $\mathbf{S}_i - \mathbf{A}_i\mathbf{S}_{N_1} - \mathbf{C}_i\mathbf{S}_{i+1}$, storing result in $\mathbf{S}_i$	$\Delta\hat{\mathbf{Q}}_i = \Delta\hat{\mathbf{Q}}'_i - \mathbf{G}_i\Delta\hat{\mathbf{Q}}_{N_1} - \mathbf{E}_i\Delta\hat{\mathbf{Q}}_{i+1}$
8	Set $\mathbf{S}_1 = \mathbf{S}_{N_1}$	$\Delta\hat{\mathbf{Q}}_1 = \Delta\hat{\mathbf{Q}}_{N_1}$

Remarks

1. The notation used in the comments in BLK4P is consistent with the notation used in the description of the algorithm in Volume 1.
2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in all three directions, the algorithm can be vectorized in one of the non-sweep directions. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

Subroutine BLK5		
Called by	Calls	Purpose
ADI	FILTER	Solve 5 x 5 block tridiagonal system of equations.

Input

A, B, C	Coefficient submatrices **A**, **B**, and **C**
NPTS	Number of grid points in the sweep direction, N.
NV	Number of grid points in the "vectorized" direction, N_v.
S	Source term subvector **S**.

Output

S	Computed solution subvector.

Description

Subroutine BLK5 solves a block tridiagonal system of equations with 5 x 5 blocks using the block matrix version of the Thomas algorithm. Subroutine FILTER is called in an attempt to eliminate any zero values on the diagonal of the submatrix **B** at the two boundaries. These can occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices **D**, **E**, and $\Delta\hat{\mathbf{Q}}'$. In Fortran, however, storage is saved by overwriting B, C, and S. The algorithm is identical to that used in subroutine BLK4. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

Remarks

1. The notation used in the comments in BLK5 is consistent with the notation used in the description of the algorithm in Volume 1.
2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in all three directions, the algorithm can be vectorized in one of the non-sweep directions. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

Subroutine BLK5P		
Called by	Calls	Purpose
ADI		Solve 5 × 5 periodic block tridiagonal system of equations.

Input

A, B, C	Coefficient submatrices **A**, **B**, and **C**
NPTS	Number of grid points in the sweep direction, N.
NV	Number of grid points in the "vectorized" direction, N_v.
S	Source term subvector **S**.

Output

S	Computed solution subvector.

Description

Subroutine BLK5P solves a periodic block tridiagonal system of equations with 5 × 5 blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices **D**, **E**, **F**, **G**, and $\Delta\hat{\mathbf{Q}}'$. In Fortran, however, storage is saved by overwriting A, B, C, and S. The algorithm is identical to that used in subroutine BLK4P. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

Remarks

1. The notation used in the comments in BLK5P is consistent with the notation used in the description of the algorithm in Volume 1.
2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in all three directions, the algorithm can be vectorized in one of the non-sweep directions. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

BLOCK DATA Subprogram		
Called by	Calls	Purpose
		Set default values for input parameters, plus a few other parameters.

Input

None.

Output

All namelist input parameters, plus:

CCP1, CCP2, CCP3, CCP4	Constants in formula for specific heat. (8.53×10^3, 3.12×10^4, 2.065×10^6, 7.83×10^8)[18]
CK1, CK2	Constants in formula for laminar thermal conductivity coefficient. (7.4907×10^{-3}, 350.0)[18]
CMU1, CMU2	Constants in formula for laminar viscosity coefficient. (7.3035×10^{-7}, 198.6)[18]
GC	Proportionality factor g_c in Newton's second law. (32.174)[18]
IBCELM	Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the ξ and/or η directions; 0 if elimination is not necessary, 1 if it is. (2*0,2*0)
IBVUP	Flags for updating boundary values from first two sweeps after third sweep; 0 if updating is not necessary, 1 if it is. (0,0,0)
ICONV	Convergence flag; 1 if converged, 0 if not. (0)
IGINT	Flags for grid interpolation requirement for the ξ, η, and ζ directions; 0 if interpolation is not necessary, 1 if it is. (0,0,0)
ITBEG	The time level n at the beginning of a run. (1)
KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic. (0,0,0)
NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations. (1,2,3,4,5)
NIN	Unit number for standard input. (5)
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T. (1,2,3,4,5)
TAU	Initial time value τ. (NTOTP*0.0)

Description

The BLOCK DATA routine is used to set default values for all the input parameters, plus various other parameters and constants. The defaults for all the input parameters are given as part of the standard input description in Section 3.1 of Volume 2. The values for the other parameters and constants set in BLOCK DATA are given in parentheses in the above output description. Note that some of these values assume

[18] These values are for reference conditions specified in English units. Values for SI units are set in subroutine INPUT.

English units are being used to specify reference conditions. If SI units are being used, these values are re-defined in subroutine INPUT.

Remarks

1. Most of the default values are defined directly, but some, like the reference viscosity MUR, are set equal to zero and defined in subroutine INPUT if not specified by the user.

Subroutine BLOUT		
Called by	Calls	Purpose
TURBBL	GATHER ISAMAX ISAMIN ISRCHFGT ISRCHFLT VORTEX WHENFLT	Compute the outer layer turbulent viscosity, using the algebraic Baldwin-Lomax model.

Input

* APLUS	Van Driest damping constant A^+.
* CB	Constant B in the Klebanoff intermittency factor.
* CCLAU	Clauser constant K in the Baldwin-Lomax outer region model.
* CCP	Constant C_{cp} in the Baldwin-Lomax outer region model.
* CKLEB, CKMIN	Constants C_{Kleb} and $(C_{Kleb})_{min}$ in the Klebanoff intermittency factor.
* CWK	Constant C_{wk} in the Baldwin-Lomax outer region model.
EP1, EP2	Minimum and maximum allowable numerical values.
* FPMIN	Value used to cut off the search for F_{max}.
* LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries.
MU	Laminar coefficient of viscosity μ_l.
NTOTP	Dimensioning parameter specifying the storage required for a full three-dimensional array (i.e., N1P × N2P × N3P).
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
N1P, N2P	Parameters specifying the dimension sizes in the ξ and η directions.
* RER	Reference Reynolds number Re_r.
RHO, U, V, W	Static density ρ, and velocities u, v, and w.
X, Y, Z	Cartesian coordinates x, y, and z.

Output

MUT	Outer layer turbulent viscosity coefficient $(\mu_t)_{outer}$.

Description

Subroutine BLOUT computes the outer layer turbulent viscosity coefficient $(\mu_t)_{outer}$ using the algebraic eddy viscosity model of Baldwin and Lomax (1978). The model is described in Section 9.1 of Volume 1. The steps performed in BLOUT are as follows:

1. Initialize the array DUMMY to zero.
2. For each grid point, set the variable DUMMY equal to a number from 1.0 to 6.0 as a flag specifying the nearest solid wall. If none of the three grid lines through the point intersect a solid wall, the point is a wake point and DUMMY = − 1.
3. Call VORTEX to compute the vorticity magnitude at each grid point.

4. If there are no wall-bounded points, skip ahead to step 9.
5. Along each grid line that intersects a solid wall, compute

$$F = y_n \left|\vec{\Omega}\right| \left(1 - e^{-y^+/A^+}\right)$$

where y_n is the distance to the wall. For each line, search outward from the wall for the first peak in F, calling it's value FPEAK. Keep searching outward, cutting off the search when F drops below FPMIN*FPEAK. (FPMIN is an input parameter with a default value of 0.9.). FPEAK is then the value of F_{max} for the current wall and grid line. Store the index corresponding to F_{max} in the LWALL parameter for the current wall and grid line.
6. At each wall-bounded grid point, compute

$$(\mu_t)_{outer} = Re_r K C_{cp} \rho F_{kleb} F_{wake}$$

In this formula,

$$F_{wake} = y_{max} F_{max}$$

where F_{max} is the appropriate value from step 5 for the nearest solid wall, and y_{max} is the corresponding value of y_n. F_{Kleb} is the Klebanoff intermittency factor, given by

$$F_{Kleb} = (C_{Kleb})_{min} + [1 - (C_{Kleb})_{min}] \left[1 + B\left(\frac{C_{Kleb}\, y_n}{y_{max}} \right)^6 \right]^{-1}$$

The Re_r in the formula for μ_t causes $(\mu_t)_{outer}$ to be nondimensionalized by μ_r.
7. The LWALL parameters used to store the indices corresponding to F_{max} are then reset to 1.
8. If there are no wake points, the calculation is finished, so skip ahead and return to the calling program.
9. For each wake point, set the variable DUMMY equal to a number from -1.0 to -6.0 as a flag specifying the nearest boundary.
10. At each grid point, compute the total velocity magnitude $\left|\vec{V}\right|$, storing it in U. The sign is set equal to the original sign of the x-velocity, for later use when U is reset to the x-velocity.
11. For each grid line, get the indices corresponding to $\left|\vec{V}\right|_{min}$ and $\left|\vec{V}\right|_{max}$.
12. For each wake point, along the grid line that intersects the nearest boundary, compute

$$F_1 = (y_n)_1 \left|\vec{\Omega}\right|$$

$$F_2 = (y_n)_2 \left|\vec{\Omega}\right|$$

where $(y_n)_1$ is the distance to the point where $\left|\vec{V}\right| = \left|\vec{V}\right|_{min}$, and $(y_n)_2$ is the distance to the point where $\left|\vec{V}\right| = \left|\vec{V}\right|_{max}$. The two values of F are stored in MUT and VORT, respectively.
13. For each grid line in the ξ direction that contains wake points, get the locations of $(F_1)_{max}$ and $(F_2)_{max}$. Since a grid line may have both wall-bounded and wake points, the F values are first gathered into a one-dimensional array containing only the wake point values. This array is then searched for the location of F_{max}, and the resulting index is converted to the proper index along the original grid line. Then, for each wake point along the grid line whose nearest boundary is at $\xi = 0$ or $\xi = 1$, compute $(y_1)_{max}$ and $(y_2)_{max}$, where $(y_1)_{max}$ is the value of $(y_n)_1$ corresponding to $(F_1)_{max}$, etc. Finally, compute

$$(\mu_t)_{outer} = K C_{cp} \rho F_{wake} Re_r$$

where

$$F_{wake} = C_{wk} V_{diff}^2 \frac{y_{max}}{F_{max}}$$

$$V_{diff} = \left| \vec{V} \right|_{max} - \left| \vec{V} \right|_{min}$$

In the computation of F_{wake}, $y_{max} = \min[(y_1)_{max}, (y_2)_{max}]$, and F_{max} is the corresponding $(F_1)_{max}$ or $(F_2)_{max}$. The Re_r in the formula for μ_t causes $(\mu_t)_{outer}$ to be nondimensionalized by μ_r. The $(\mu_t)_{outer}$ values are stored in DUMMY so the loop will vectorize. They are flagged as wake point values by making them negative.

14. Repeat step 13 for grid lines in the η and ζ directions.
15. For each wake grid point, move $(\mu_t)_{outer}$ from DUMMY into MUT, making it positive.
16. Reset U to the value of the x-velocity.

Remarks

1. To avoid the possibility of floating point errors, the values of $\left|\vec{V}_{max}\right|$, $\left|\vec{V}_{min}\right|$, and F_{max} are set to a minimum of 10^{-10}.

Subroutine BVUP		
Called by	**Calls**	**Purpose**
EXEC	BCGEN EQSTAT SGEFA SGESL	Update first and second sweep boundary values after third sweep.

Input

	DXI, DETA	Computational grid spacing $\Delta\xi$ and $\Delta\eta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
	IBVUP	Flags for updating boundary values from first two sweeps after third sweep; 0 if updating is not necessary, 1 if it is.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
	NEQ	Number of coupled equations being solved, N_{eq}.
	NEQP	Dimensioning parameter specifying maximum number of coupled equations allowed.
	NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, N_1+1, N_2+1, and N_3+1 for spatially periodic boundary condition in ξ, η, and ζ.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	N1P, N2P	Parameters specifying the dimension sizes in the ξ and η directions.
	RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n at all grid points.
	RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level $n+1$ at all interior grid points.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.

Output

DEL	Computational grid spacing for the sweep direction being updated.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
ISWEEP	ADI sweep number for sweep direction being updated.
IV	Index in the "vectorized" direction, i_v.
METX, METY, METZ, METT	Derivatives of computational coordinate, for the sweep direction being updated, with respect to x, y, z, and t.
NPTS	Number of grid points N in the sweep direction being updated.

NV	Number of grid points in the "vectorized" direction, N_v.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level $n+1$ at boundary points from first and second sweep.

Description

Subroutine BVUP updates boundary values from the first and second, or ξ and η, sweeps after the third, or ζ, sweep. In general, this is necessary when gradient or extrapolation boundary conditions are used in the ξ or η direction. Some updating is also necessary when spatially periodic boundary conditions are used. The procedure for non-periodic boundary conditions is described in Section 7.3 of Volume 1.

Updating boundary values is complicated somewhat when spatially periodic boundary conditions are used.

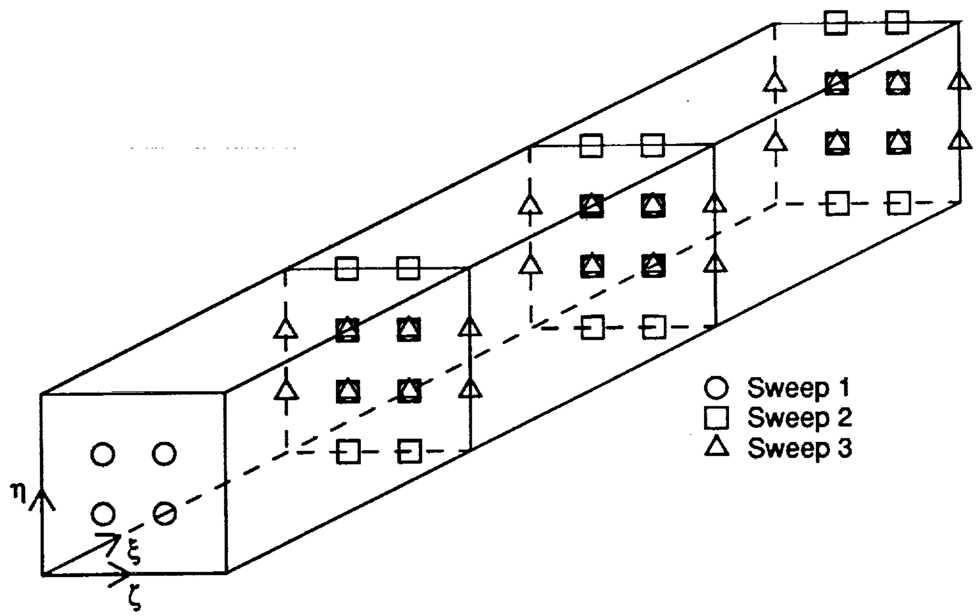

Figure 4.1 - Updating boundary values for periodic boundary conditions in the ξ direction.

The situation for a periodic boundary condition in the ξ direction but not in the η or ζ directions is shown in Figure 4.1. In the figure, a $4 \times 4 \times 4$ grid is shown in computational space for a three-dimensional problem. The circles and squares represent grid points at which intermediate values are computed during the first two ADI sweeps, and the triangles represent grid points at which final values are computed during the third ADI sweep. The intermediate values at $\eta = 0$ and at $\eta = 1$ are updated first. This is done using the same procedure as for non-periodic boundary conditions, described in Section 7.3 of Volume 1, but for $i = 2$ to N_1 instead of $N_1 - 1$, where i is the index in the ξ direction. The values on the η-ζ edges (i.e., the four lines of intersection between the η and ζ boundary planes) are also updated over the same ξ indices using the procedure described in Section 7.3 of Volume 1. Finally, the values in the $\xi = 0$ plane are updated by setting $\hat{Q}_1 = \hat{Q}_{N_1}$ at every point in the plane.

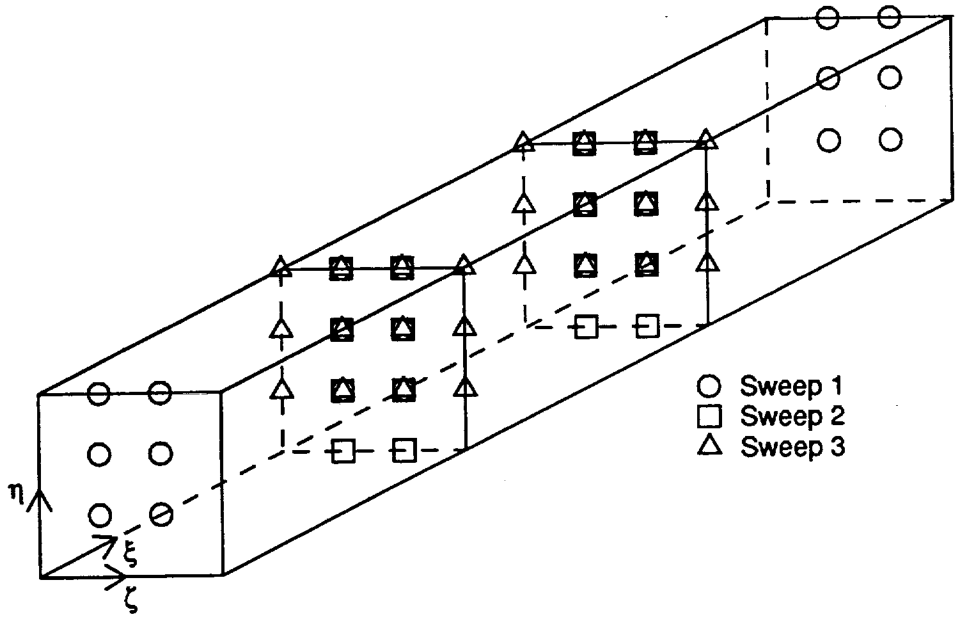

Figure 4.2 - Updating boundary values for periodic boundary conditions in the η direction.

The situation for a periodic boundary condition in the η direction but not in the ξ or ζ directions is shown in Figure 4.2. In this case, the intermediate values at $\xi = 0$ and at $\xi = 1$ are updated first. This is done using the same procedure as for non-periodic boundary conditions, described in Section 7.3 of Volume 1, but for $j = 2$ to N_2 instead of $N_2 - 1$, where j is the index in the η direction. The values on the ξ-ζ edges (i.e., the four lines of intersection between the ξ and ζ boundary planes) are also updated over the same η indices using the procedure described in Section 7.3 of Volume 1. Finally, the values in the $\eta = 0$ plane are updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_2}$ at every point in the plane.

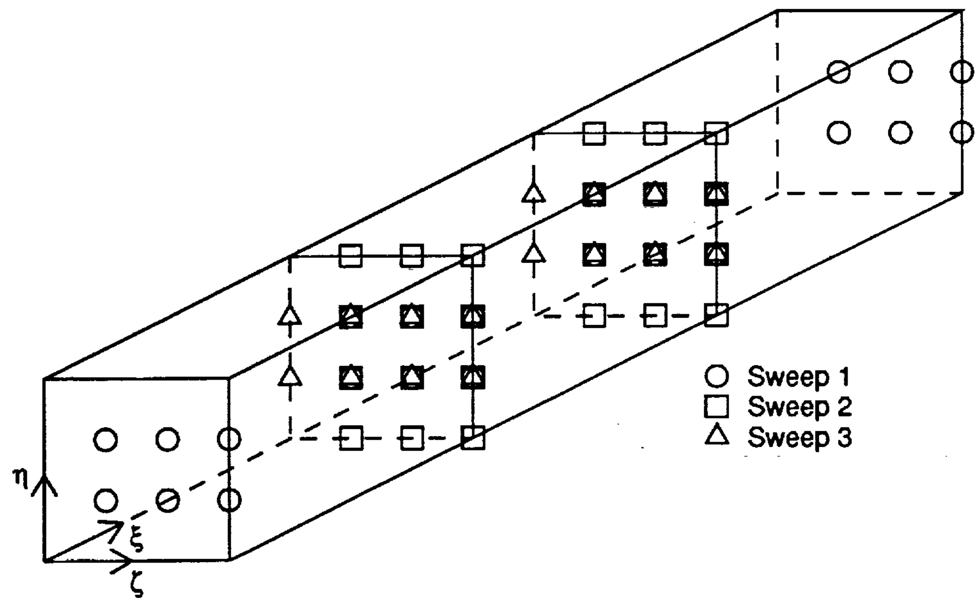

Figure 4.3 - Updating boundary values for periodic boundary conditions in the ζ direction.

The situation for a periodic boundary condition in the ζ direction but not in the ξ or η directions is shown in Figure 4.3. In this case, the intermediate values at $\xi = 0$, $\xi = 1$, $\eta = 0$, and $\eta = 1$ are updated first. This is done using the same procedure as for non-periodic boundary conditions, described in Section 7.3 of Volume 1, but for $k = 2$ to N_3 instead of $N_3 - 1$, where k is the index in the ζ direction. The values on the ξ-η edges (i.e., the four lines of intersection between the ξ and η boundary planes) are also updated over the same ζ indices using the procedure described in Section 7.3 of Volume 1. Finally, the values in the $\zeta = 0$ plane are updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_3}$ at every point in the plane.

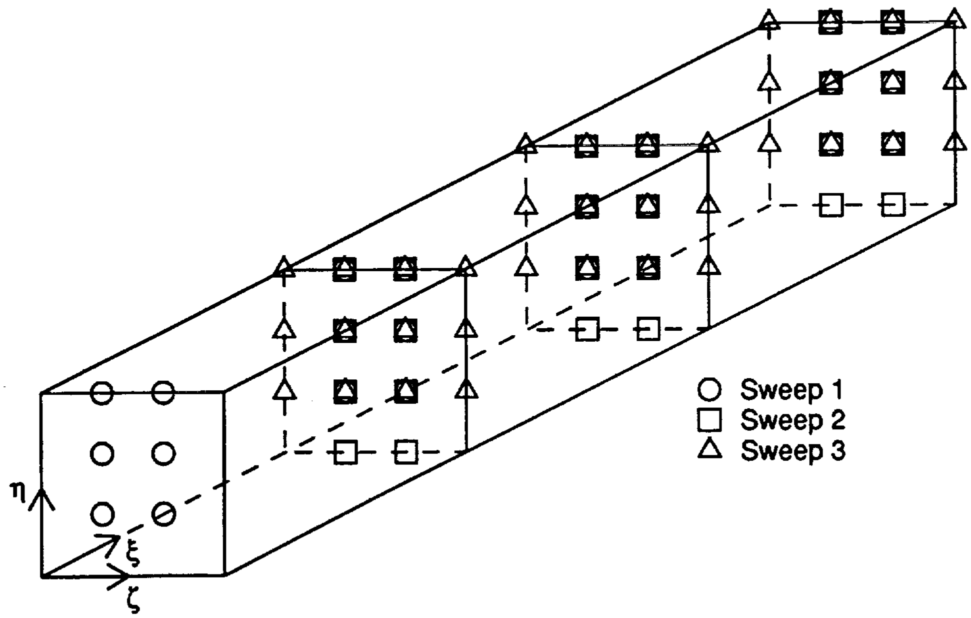

Figure 4.4 - Updating boundary values for periodic boundary conditions in the ξ and η directions.

The situation for periodic boundary conditions in the ξ and η directions but not in the ζ direction is shown in Figure 4.4. In this case, the only action needed is to update the values in the $\xi = 0$ and $\eta = 0$ planes, by setting $\hat{Q}_1 = \hat{Q}_{N_1}$ at every point in the $\xi = 0$ plane, and $\hat{Q}_1 = \hat{Q}_{N_2}$ at every point in the $\eta = 0$ plane.

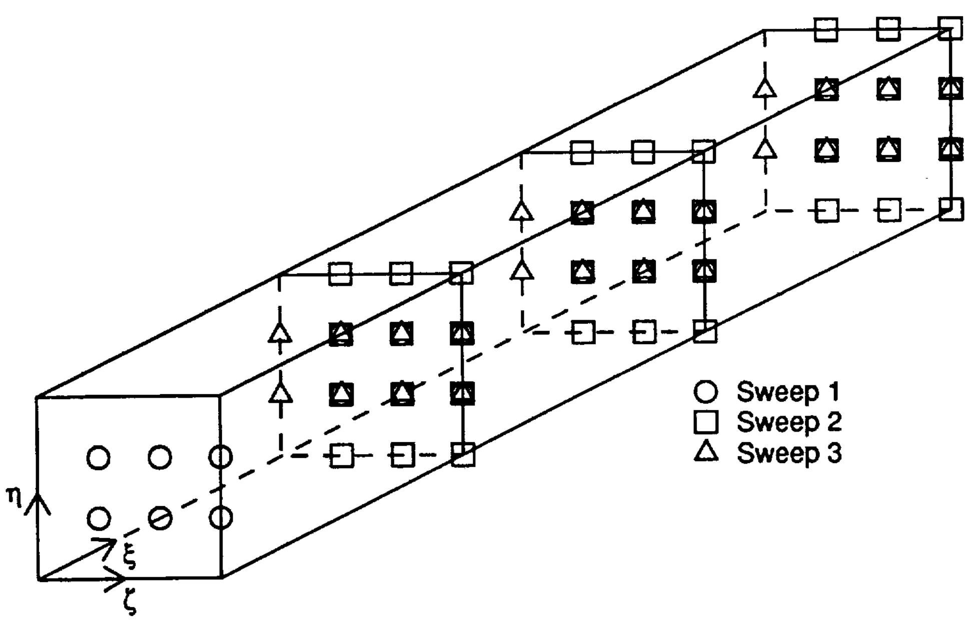

Figure 4.5 - Updating boundary values for periodic boundary conditions in the ξ and ζ directions.

The situation for periodic boundary conditions in the ξ and ζ directions but not in the η direction is shown in Figure 4.5. In this case, the intermediate values at $\eta = 0$ and at $\eta = 1$ are updated first. This is done using the same procedure as for non-periodic boundary conditions, described in Section 7.3 of Volume 1, but for $i = 2$ to N_1 instead of $N_1 - 1$, and for $k = 2$ to N_3 instead of $N_3 - 1$, where i and k are the indices in the ξ and ζ directions. The values in the $\xi = 0$ plane are then updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_1}$ at every point in the plane except for the points $(1, j, 1)$. Finally, the remaining points in the $\zeta = 0$ plane are updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_3}$.

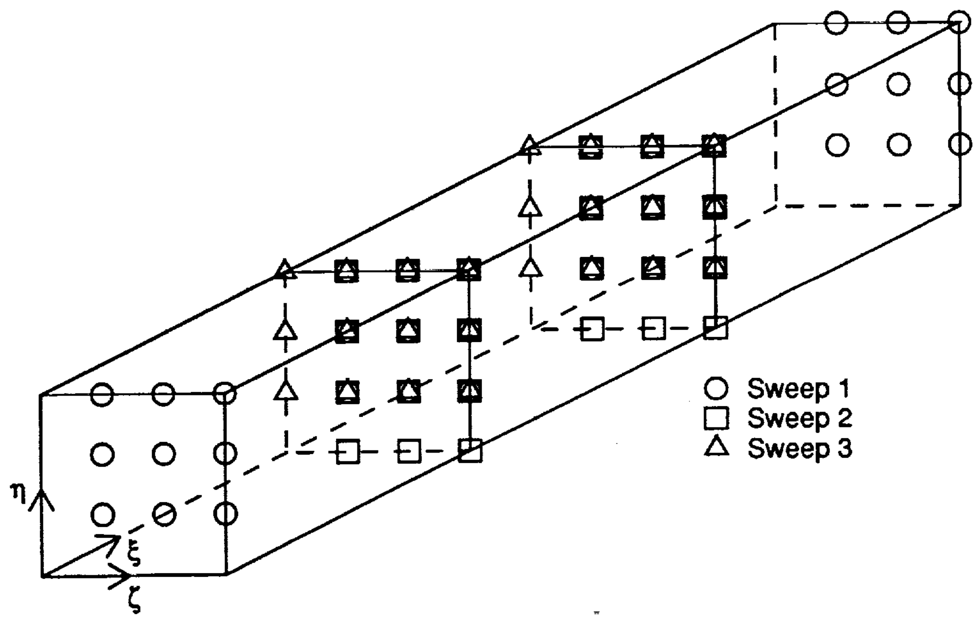

Figure 4.6 - Updating boundary values for periodic boundary conditions in the η and ζ directions.

The situation for periodic boundary conditions in the η and ζ directions but not in the ξ direction is shown in Figure 4.6. In this case, the intermediate values at $\xi = 0$ and at $\xi = 1$ are updated first. This is done using the same procedure as for non-periodic boundary conditions, described in Section 7.3 of Volume 1, but for $j = 2$ to N_2 instead of $N_2 - 1$, and for $k = 2$ to N_3 instead of $N_3 - 1$, where j and k are the indices in the η and ζ directions. The values in the $\eta = 0$ plane are then updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_2}$ at every point in the plane except for the points $(i, 1, 1)$. Finally, the remaining points in the $\zeta = 0$ plane are updated by setting $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_{N_3}$.

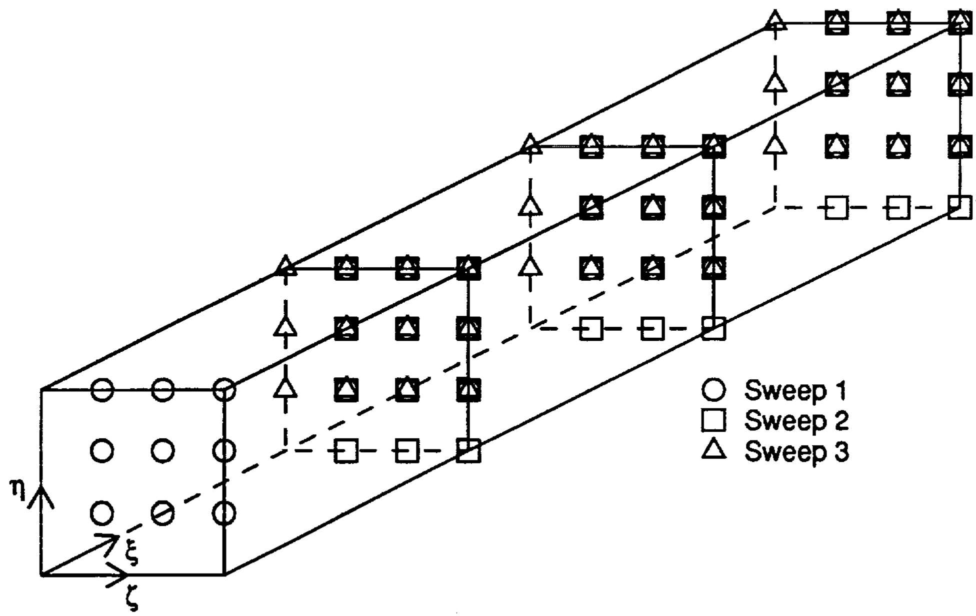

Figure 4.7 - Updating boundary values for periodic boundary conditions in all three directions.

The situation for periodic boundary conditions in all three coordinate directions is shown in Figure 4.7. In this case, the only action needed is to update the values in the $\xi = 0$ and $\eta = 0$ planes, by setting $\hat{Q}_1 = \hat{Q}_{N_1}$ in the $\xi = 0$ plane and $\hat{Q}_1 = \hat{Q}_{N_2}$ in the $\eta = 0$ plane.

<u>**Remarks**</u>

1. The corner values of ρ and E_T are updated by linearly extrapolating from the two adjacent points in the ξ, η, and ζ directions, and averaging the three results. Note that this extrapolation is done in computational space. Grid packing in any direction is thus not taken into account. The corner values of the velocities are updated by doing the same type of extrapolation. Instead of averaging, however, the extrapolated velocity whose absolute value is lower is used. This was done to maintain no-slip at duct inlets and exits.
2. Subroutines SGEFA and SGESL are Cray LINPACK routines. In general terms, if the Fortran arrays A and S represent **A** and **S**, where **A** is a square N by N matrix and **S** is a vector with N elements, and if the leading dimension of the Fortran array A is LDA, then the Fortran sequence

```
call sgefa (a,lda,n,ipvt,info)
call sgesl (a,lda,n,ipvt,s,0)
```

computes $\mathbf{A}^{-1}\mathbf{S}$, storing the result in S.

Subroutine COEFC		
Called by	Calls	Purpose
EXEC		Compute coefficients and source term for the continuity equation.

Input

DEL	Computational grid spacing in sweep direction.
DTAU	Time step $\Delta\tau$.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
* IHSTAG	Flag for constant stagnation enthalpy option.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I2, I3	Grid indices j and k, in the η and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
NC	Array index associated with the continuity equation.
NPTS	Number of grid points in the sweep direction, N.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.
RHOL	Static density ρ from previous ADI sweep.
* THC	Parameters θ_1 and θ_2 determining type of time differencing for the continuity equation.
XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at interior points (row NC only).
S	Source term subvector **S** at interior points (element NC only).

Description

Subroutine COEFC computes the coefficients and source term for the continuity equation. Equations (7.5a-c) in Volume 1 represent, in vector form, the five governing difference equations for the three ADI sweeps. The elements of the inviscid flux vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{\mathbf{E}}_{V_1}$, $\hat{\mathbf{E}}_{V_2}$, etc., are given in Appendix A of Volume 1. The Jacobian

coefficient matrices $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$, $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the continuity equation may be written for the three ADI sweeps as[19]

<u>Sweep 1 (ξ direction)</u>

$$\Delta\hat{\rho}_i^* + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\xi}\left[\left(\frac{\partial \hat{\mathbf{E}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{i+1} \Delta\hat{\mathbf{Q}}^*_{i+1} - \left(\frac{\partial \hat{\mathbf{E}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{i-1}\Delta\hat{\mathbf{Q}}^*_{i-1}\right] = -\frac{\Delta\tau}{1+\theta_2}\left(\delta_\xi \hat{\mathbf{E}}_1 + \delta_\eta \hat{\mathbf{F}}_1 + \delta_\zeta \hat{\mathbf{G}}_1\right)^n + \frac{\theta_2}{1+\theta_2}\Delta\hat{\rho}^{n-1}$$

<u>Sweep 2 (η direction)</u>

$$\Delta\hat{\rho}_j^{**} + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\eta}\left[\left(\frac{\partial \hat{\mathbf{F}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{j+1} \Delta\hat{\mathbf{Q}}^{**}_{j+1} - \left(\frac{\partial \hat{\mathbf{F}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{j-1}\Delta\hat{\mathbf{Q}}^{**}_{j-1}\right] = \Delta\hat{\rho}^*$$

<u>Sweep 3 (ζ direction)</u>

$$\Delta\hat{\rho}_k^{n} + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\zeta}\left[\left(\frac{\partial \hat{\mathbf{G}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{k+1} \Delta\hat{\mathbf{Q}}^{n}_{k+1} - \left(\frac{\partial \hat{\mathbf{G}}_1}{\partial \hat{\mathbf{Q}}}\right)^n_{k-1}\Delta\hat{\mathbf{Q}}^{n}_{k-1}\right] = \Delta\hat{\rho}^{**}$$

In the above equations, the subscripts i, j, and k represent grid point indices in the ξ, η, and ζ directions. For notational convenience, terms without an explicitly written i, j, or k subscript are understood to be at i, j, or k.

The vector of dependent variables is

$$\hat{\mathbf{Q}} = \frac{1}{J}\left[\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T\right]^T$$

The appropriate elements of the flux vectors are given by

$$\hat{\mathbf{E}}_1 = \frac{1}{J}\left[\rho u\xi_x + \rho v\xi_y + \rho w\xi_z + \rho\xi_t\right]$$

$$\hat{\mathbf{F}}_1 = \frac{1}{J}\left[\rho u\eta_x + \rho v\eta_y + \rho w\eta_z + \rho\eta_t\right]$$

$$\hat{\mathbf{G}}_1 = \frac{1}{J}\left[\rho u\zeta_x + \rho v\zeta_y + \rho w\zeta_z + \rho\zeta_t\right]$$

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$ for the continuity equation are

---

19 These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy E_T would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.

$$\frac{\partial \hat{\mathbf{E}}_1}{\partial \hat{\mathbf{Q}}} = [\xi_t \quad \xi_x \quad \xi_y \quad \xi_z \quad 0]$$

The Jacobian coefficient matrices $\partial \hat{\mathbf{F}}_1/\partial \hat{\mathbf{Q}}$ and $\partial \hat{\mathbf{G}}_1/\partial \hat{\mathbf{Q}}$ have the same form as $\partial \hat{\mathbf{E}}_1/\partial \hat{\mathbf{Q}}$, but with ξ replaced by η and ζ, respectively.

As an example of how these equations are translated into Fortran, consider the $\Delta(\rho u/J)$ term on the left hand side for the first sweep. This is the second element of $\hat{\mathbf{Q}}$, so using the second element in $\partial \hat{\mathbf{E}}_1/\partial \hat{\mathbf{Q}}$ we get

$$\mathrm{A(IV,I,NC,NRU)} = -\frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}(\xi_x)_{i-1,j,k}$$

$$\mathrm{B(IV,I,NC,NRU)} = 0$$

$$\mathrm{C(IV,I,NC,NRU)} = \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}(\xi_x)_{i+1,j,k}$$

In COEFC, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second and third ADI sweeps have exactly the same form as for the first sweep, but with ξ replaced by η and ζ, respectively. By defining DEL, METX, METY, METZ, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for all three sweeps.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1. This is followed by the coding for the source term for the second and third sweeps, which consists only of $\Delta\hat{\rho}^*$ or $\Delta\hat{\rho}^{**}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply to all three sweeps (i.e., the implicit terms), the first two subscripts are written as (IV,I). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.

Subroutine COEFE1		
Called by	Calls	Purpose
EXEC		Compute coefficients and first-sweep source term, except for cross-derivative viscous terms, for the energy equation.

Input

	DEL	Computational grid spacing in sweep direction.
	DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial \rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
	DTAU	Time step $\Delta\tau$.
	DTDRHO, DTDRU, DTDRV, DTDRW, DTDET	Derivatives $\partial T/\partial \rho$, $\partial T/\partial(\rho u)$, $\partial T/\partial(\rho v)$, $\partial T/\partial(\rho w)$, and $\partial T/\partial E_T$.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
	ETL	Total energy E_T from previous ADI sweep.
	IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
*	IEULER	Flag for Euler calculation.
	ISWEEP	Current ADI sweep number.
*	ITHIN	Flags for thin-layer option.
	IV	Index in the "vectorized" direction, i_v.
	I2, I3	Grid indices j and k, in the η and ζ directions.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
	MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k at time level n.
	NEN	Array index associated with the energy equation.
	NPTS	Number of grid points in the sweep direction, N.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
	P, T	Static pressure p and temperature T at time level n.
	PRR	Reference Prandtl number Pr_r.
*	RER	Reference Reynolds number Re_r.
	RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
*	THE	Parameters θ_1, θ_2, and θ_3 determining type of time differencing for the energy equation.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.

ZETAX, ZETAY, ZETAZ, ZETAT — Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

A, B, C — Coefficient submatrices **A**, **B**, and **C** at interior points (row NEN only).

S — First-sweep source term subvector **S** at interior points, except for the cross-derivative viscous terms (element NEN only).

Description

Subroutine COEFE1 computes the coefficients and starts the computation of the first-sweep source term for the energy equation. The cross-derivative viscous terms are added to the first-sweep source term in subroutine COEFE2. Equations (7.5a-c) in Volume 1 represent, in vector form, the five governing difference equations for the three ADI sweeps. The elements of the inviscid flux vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{\mathbf{E}}_{V_1}$, $\hat{\mathbf{E}}_{V_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$, $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the energy equation may be written for the three ADI sweeps as

Sweep 1 (ξ direction)

$$\Delta(\hat{E}_T)_i^* + \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\xi}\left[\left(\frac{\partial\hat{\mathbf{E}}_5}{\partial\hat{\mathbf{Q}}}\right)_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^* - \left(\frac{\partial\hat{\mathbf{E}}_5}{\partial\hat{\mathbf{Q}}}\right)_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^*\right]$$

$$- \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\xi)^2}\left[(f_{i-1}+f_i)^n g_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^* - (f_{i-1}+2f_i+f_{i+1})^n g_i^n \Delta\hat{\mathbf{Q}}_i^* + (f_i+f_{i+1})^n g_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^*\right] =$$

$$- \frac{\Delta\tau}{1+\theta_2}\left(\delta_\xi\hat{\mathbf{E}}_5 + \delta_\eta\hat{\mathbf{F}}_5 + \delta_\zeta\hat{\mathbf{G}}_5\right)^n + \frac{\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_1})_5 + \delta_\eta(\hat{\mathbf{F}}_{V_1})_5 + \delta_\zeta(\hat{\mathbf{G}}_{V_1})_5\right]^n$$

$$+ \frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_5 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_5 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_5\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_5 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_5 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_5\right]^{n-1}$$

$$+ \frac{\theta_2}{1+\theta_2}\Delta\hat{E}_T^{n-1}$$

Sweep 2 (η direction)

$$\Delta(\hat{E}_T)_j^{**} + \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\eta}\left[\left(\frac{\partial\hat{\mathbf{F}}_5}{\partial\hat{\mathbf{Q}}}\right)_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**} - \left(\frac{\partial\hat{\mathbf{F}}_5}{\partial\hat{\mathbf{Q}}}\right)_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**}\right]$$

$$- \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\eta)^2}\left[(f_{j-1}+f_j)^n g_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**} - (f_{j-1}+2f_j+f_{j+1})^n g_j^n \Delta\hat{\mathbf{Q}}_j^{**} + (f_j+f_{j+1})^n g_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**}\right] =$$

$$\Delta\hat{E}_T^*$$

Sweep 3 (ζ direction)

$$\Delta(\hat{E}_T)_k^n + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\zeta}\left[\left(\frac{\partial \hat{\mathbf{G}}_5}{\partial \hat{\mathbf{Q}}}\right)_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n - \left(\frac{\partial \hat{\mathbf{G}}_5}{\partial \hat{\mathbf{Q}}}\right)_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n\right]$$

$$- \frac{\theta_1 \Delta\tau}{(1+\theta_2)2(\Delta\zeta)^2}\left[(f_{k-1}+f_k)^n g_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n - (f_{k-1}+2f_k+f_{k+1})^n g_k^n \Delta\hat{\mathbf{Q}}_k^n + (f_k+f_{k+1})^n g_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n\right] =$$

$$\Delta\hat{E}_T^{**}$$

In the above equations, the subscripts i, j, and k represent grid point indices in the ξ, η, and ζ directions. For notational convenience, terms without an explicitly written i, j, or k subscript are understood to be at i, j, or k. On the left hand side, f is the coefficient of $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$, depending on the sweep) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix. Similarly, g is the term in the parentheses following $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{\mathbf{Q}} = \frac{1}{J}[\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{\mathbf{E}}_5 = \frac{1}{J}[(E_T+p)u\xi_x + (E_T+p)v\xi_y + (E_T+p)w\xi_z + E_T\xi_t]$$

$$\hat{\mathbf{F}}_5 = \frac{1}{J}[(E_T+p)u\eta_x + (E_T+p)v\eta_y + (E_T+p)w\eta_z + E_T\eta_t]$$

$$\hat{\mathbf{G}}_5 = \frac{1}{J}[(E_T+p)u\zeta_x + (E_T+p)v\zeta_y + (E_T+p)w\zeta_z + E_T\zeta_t]$$

The approp..ate elements of the non-cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_1})_5 = \frac{1}{J}\frac{1}{Re_r}\left\{\frac{(2\mu+\lambda)}{2}[\xi_x^2(u^2)_\xi + \xi_y^2(v^2)_\xi + \xi_z^2(w^2)_\xi] + (\mu+\lambda)[\xi_x\xi_y(uv)_\xi + \xi_x\xi_z(uw)_\xi + \xi_y\xi_z(vw)_\xi]\right.$$
$$\left. + \frac{\mu}{2}[\xi_x^2(v^2+w^2)_\xi + \xi_y^2(u^2+w^2)_\xi + \xi_z^2(u^2+v^2)_\xi] + \frac{k}{Pr_r}(\xi_x^2+\xi_y^2+\xi_z^2)T_\xi\right\}$$

$$(\hat{\mathbf{F}}_{V_1})_5 = \frac{1}{J}\frac{1}{Re_r}\left\{\frac{(2\mu+\lambda)}{2}[\eta_x^2(u^2)_\eta + \eta_y^2(v^2)_\eta + \eta_z^2(w^2)_\eta] + (\mu+\lambda)[\eta_x\eta_y(uv)_\eta + \eta_x\eta_z(uw)_\eta + \eta_y\eta_z(vw)_\eta]\right.$$
$$\left. + \frac{\mu}{2}[\eta_x^2(v^2+w^2)_\eta + \eta_y^2(u^2+w^2)_\eta + \eta_z^2(u^2+v^2)_\eta] + \frac{k}{Pr_r}(\eta_x^2+\eta_y^2+\eta_z^2)T_\eta\right\}$$

$$(\hat{\mathbf{G}}_{V_1})_5 = \frac{1}{J}\frac{1}{Re_r}\left\{\frac{(2\mu+\lambda)}{2}[\zeta_x^2(u^2)_\zeta + \zeta_y^2(v^2)_\zeta + \zeta_z^2(w^2)_\zeta] + (\mu+\lambda)[\zeta_x\zeta_y(uv)_\zeta + \zeta_x\zeta_z(uw)_\zeta + \zeta_y\zeta_z(vw)_\zeta]\right.$$
$$\left. + \frac{\mu}{2}[\zeta_x^2(v^2+w^2)_\zeta + \zeta_y^2(u^2+w^2)_\zeta + \zeta_z^2(u^2+v^2)_\zeta] + \frac{k}{Pr_r}(\zeta_x^2+\zeta_y^2+\zeta_z^2)T_\zeta\right\}$$

The terms involving $\hat{\mathbf{E}}_{V_2}$, $\hat{\mathbf{F}}_{V_2}$, and $\hat{\mathbf{G}}_{V_2}$ are the cross derivative viscous source terms, and are computed in subroutine COEFE2.

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$ for the inviscid terms in the energy equation are

$$\frac{\partial\hat{\mathbf{E}}_5}{\partial\hat{\mathbf{Q}}} = \left[-f_1\left(f_2 - \frac{\partial p}{\partial\rho}\right) \quad f_2\xi_x + f_1\frac{\partial p}{\partial(\rho u)} \quad f_2\xi_y + f_1\frac{\partial p}{\partial(\rho v)} \quad f_2\xi_z + f_1\frac{\partial p}{\partial(\rho w)} \quad \xi_t + f_1\left(1 + \frac{\partial p}{\partial E_T}\right)\right]$$

where $f_1 = u\xi_x + v\xi_y + w\xi_z$ and $f_2 = (E_T + p)/\rho$.

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ for the viscous terms are

$$\frac{\partial(\hat{\mathbf{E}}_{V_1})_5}{\partial\hat{\mathbf{Q}}} = \frac{1}{Re_r}\left[\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{51} \quad \left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{52} \quad \left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{53} \quad \left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{54} \quad \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial E_T}\right)\right]$$

where

$$\begin{aligned}
\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{51} &= -\alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{u^2}{\rho}\right) - \alpha_{yy}\frac{\partial}{\partial\xi}\left(\frac{v^2}{\rho}\right) - \alpha_{zz}\frac{\partial}{\partial\xi}\left(\frac{w^2}{\rho}\right) \\
&\quad - 2\alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{uv}{\rho}\right) - 2\alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{uw}{\rho}\right) - 2\alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{vw}{\rho}\right) + \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial\rho}\right)
\end{aligned}$$

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{52} = \alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) + \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) + \alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right) + \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial(\rho u)}\right)$$

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{53} = \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) + \alpha_{yy}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) + \alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right) + \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial(\rho v)}\right)$$

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{54} = \alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) + \alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) + \alpha_{zz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right) + \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial(\rho v)}\right)$$

$$\begin{aligned}
\alpha_{xx} &= (2\mu + \lambda){\xi_x}^2 + \mu{\xi_y}^2 + \mu{\xi_z}^2 \\
\alpha_{yy} &= \mu{\xi_x}^2 + (2\mu + \lambda){\xi_y}^2 + \mu{\xi_z}^2 \\
\alpha_{zz} &= \mu{\xi_x}^2 + \mu{\xi_y}^2 + (2\mu + \lambda){\xi_z}^2 \\
\alpha_{xy} &= (\mu + \lambda)\xi_x\xi_y \\
\alpha_{xz} &= (\mu + \lambda)\xi_x\xi_z \\
\alpha_{yz} &= (\mu + \lambda)\xi_y\xi_z \\
\alpha_0 &= \frac{k}{Pr_r}({\xi_x}^2 + {\xi_y}^2 + {\xi_z}^2)
\end{aligned}$$

The Jacobian coefficient matrices $\partial\hat{\mathbf{F}}_5/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{F}}_{V_1})_5/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_5/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_5/\partial\hat{\mathbf{Q}}$, but with ξ replaced by η. Similarly, the Jacobian coefficient matrices $\partial\hat{\mathbf{G}}_5/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{G}}_{V_1})_5/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_5/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_5/\partial\hat{\mathbf{Q}}$, but with ξ replaced by ζ.

As an example of how these equations are translated into Fortran, consider the $\Delta(\rho u/J)$ term on the left hand side for the first sweep. This is the second element of $\hat{\mathbf{Q}}$, so using the second element in $\partial\hat{\mathbf{E}}_5/\partial\hat{\mathbf{Q}}$ we get for the inviscid term

$$\text{A(IV,I,NEN,NRU)} = -\frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left\{\left(\frac{E_T+p}{\rho}\xi_x\right)_{i-1,j,k} + \left[(u\xi_x + v\xi_y + w\xi_z)\frac{\partial p}{\partial(\rho u)}\right]_{i-1,j,k}\right\}$$

$$\text{B(IV,I,NEN,NRU)} = 0$$

$$\text{C(IV,I,NEN,NRU)} = \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left\{\left(\frac{E_T+p}{\rho}\xi_x\right)_{i+1,j,k} + \left[(u\xi_x + v\xi_y + w\xi_z)\frac{\partial p}{\partial(\rho u)}\right]_{i+1,j,k}\right\}$$

For the viscous terms on the left hand side, we use the second element in $\partial(\hat{\mathbf{E}}_{V_1})_5/\partial\hat{\mathbf{Q}}$, which is

$$\frac{1}{Re_r}\left[\alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) + \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) + \alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right) + \alpha_0\frac{\partial}{\partial\xi}\left(\frac{\partial T}{\partial(\rho u)}\right)\right]$$

There are four terms in that element. Thus, in turn, $f = \alpha_{xx}/Re_r$, α_{xy}/Re_r, α_{xz}/Re_r, and α_0/Re_r, and $g = u/\rho$, v/ρ, w/ρ, and $\partial T/\partial(\rho u)$. To add the viscous contribution to this part of the **A** coefficient submatrix, we therefore set

$$\begin{aligned}
&\text{A(IV,I,NEN,NRU)} = \text{A(IV,I,NEN,NRU)} \\
&- \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2(\Delta\xi)^2 Re_r}\left\{[(\alpha_{xx})_{i-1,j,k} + (\alpha_{xx})_{i,j,k}]\left(\frac{u}{\rho}\right)_{i-1,j,k} + [(\alpha_{xy})_{i-1,j,k} + (\alpha_{xy})_{i,j,k}]\left(\frac{v}{\rho}\right)_{i-1,j,k}\right. \\
&\left. + [(\alpha_{xz})_{i-1,j,k} + (\alpha_{xz})_{i,j,k}]\left(\frac{w}{\rho}\right)_{i-1,j,k} + [(\alpha_0)_{i-1,j,k} + (\alpha_0)_{i,j,k}]\left(\frac{\partial T}{\partial(\rho u)}\right)_{i-1,j,k}\right\}
\end{aligned}$$

Similar equations may be written for the **B** and **C** coefficient submatrices.

In COEFE1, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second and third ADI sweeps have exactly the same form as for the first sweep, but with ξ replaced by η and ζ, respectively. By defining DEL, METX, METY, METZ, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for all three sweeps.

The non-cross-derivative part of the source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply to all three sweeps (i.e., the implicit terms), the first two subscripts are written as (IV,I). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.

3. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous and heat conduction terms.
4. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous and heat conduction terms containing derivatives in the specified direction.
5. The computation of the first-sweep source term was split in two to keep the energy equation subroutine from being even longer than it already is.

Subroutine COEFE2		
Called by	Calls	Purpose
EXEC		Compute cross-derivative part of first-sweep source term, plus second- and third-sweep source terms, for the energy equation.

Input

DTAU	Time step $\Delta\tau$.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
* IEULER	Flag for Euler calculation.
ISWEEP	Current ADI sweep number.
* ITHIN	Flags for thin-layer option.
I2, I3	Grid indices j and k, in the η and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k at time level n.
NEN	Array index associated with the energy equation.
NPTS	Number of grid points in the sweep direction, N.
T	Static temperature T at time level n.
PRR	Reference Prandtl number Pr_r.
* RER	Reference Reynolds number Re_r.
S	First-sweep source term subvector **S** at interior points, except for the cross-derivative viscous terms (element NEN only).
* THE	Parameters θ_1, θ_2, and θ_3 determining type of time differencing for the energy equation.
TL	Static temperature T from previous ADI sweep.
U, V, W, ET	Velocities u, v, and w, and total energy E_T at time level n.
UL, VL, WL, ETL	Velocities u, v, and w, and total energy E_T from previous ADI sweep.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

S	Source term subvector **S** at interior points (element NEN only).

Description

Subroutine COEFE2 finishes the computation of the first-sweep source term for the energy equation by adding the cross-derivative terms. It also computes the second- and third-sweep source terms. The

differenced form of the energy equation for the three ADI sweeps is presented in the description of subroutine COEFE1.

The appropriate elements of the cross derivative viscous flux vectors are

$$
\begin{aligned}
(\hat{\mathbf{E}}_{V_2})_5 = \frac{1}{J}\frac{1}{Re_r}\Big\{ & 2\mu[\xi_x(\eta_x u u_\eta + \zeta_x u u_\zeta) + \xi_y(\eta_y v v_\eta + \zeta_y v v_\zeta) + \xi_z(\eta_z w w_\eta + \zeta_z w w_\zeta)] \\
& + \lambda\xi_x(\eta_x u u_\eta + \eta_y u v_\eta + \eta_z u w_\eta + \zeta_x u u_\zeta + \zeta_y u v_\zeta + \zeta_z u w_\zeta) \\
& + \lambda\xi_y(\eta_x v u_\eta + \eta_y v v_\eta + \eta_z v w_\eta + \zeta_x v u_\zeta + \zeta_y v v_\zeta + \zeta_z v w_\zeta) \\
& + \lambda\xi_z(\eta_x w u_\eta + \eta_y w v_\eta + \eta_z w w_\eta + \zeta_x w u_\zeta + \zeta_y w v_\zeta + \zeta_z w w_\zeta) \\
& + \mu\xi_x(\eta_y v u_\eta + \eta_x v v_\eta + \eta_z w u_\eta + \eta_x w w_\eta + \zeta_y v u_\zeta + \zeta_x v v_\zeta + \zeta_z w u_\zeta + \zeta_x w w_\zeta) \\
& + \mu\xi_y(\eta_y u u_\eta + \eta_x u v_\eta + \eta_z w v_\eta + \eta_y w w_\eta + \zeta_y u u_\zeta + \zeta_x u v_\zeta + \zeta_z w v_\zeta + \zeta_y w w_\zeta) \\
& + \mu\xi_z(\eta_z u u_\eta + \eta_x u w_\eta + \eta_z v v_\eta + \eta_y v w_\eta + \zeta_z u u_\zeta + \zeta_x u w_\zeta + \zeta_z v v_\zeta + \zeta_y v w_\zeta) \\
& + \frac{k}{Pr_r}(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z)T_\eta + \frac{k}{Pr_r}(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)T_\zeta \Big\}
\end{aligned}
$$

$$
\begin{aligned}
(\hat{\mathbf{F}}_{V_2})_5 = \frac{1}{J}\frac{1}{Re_r}\Big\{ & 2\mu[\eta_x(\xi_x u u_\xi + \zeta_x u u_\zeta) + \eta_y(\xi_y v v_\xi + \zeta_y v v_\zeta) + \eta_z(\xi_z w w_\xi + \zeta_z w w_\zeta)] \\
& + \lambda\eta_x(\xi_x u u_\xi + \xi_y u v_\xi + \xi_z u w_\xi + \zeta_x u u_\zeta + \zeta_y u v_\zeta + \zeta_z u w_\zeta) \\
& + \lambda\eta_y(\xi_x v u_\xi + \xi_y v v_\xi + \xi_z v w_\xi + \zeta_x v u_\zeta + \zeta_y v v_\zeta + \zeta_z v w_\zeta) \\
& + \lambda\eta_z(\xi_x w u_\xi + \xi_y w v_\xi + \xi_z w w_\xi + \zeta_x w u_\zeta + \zeta_y w v_\zeta + \zeta_z w w_\zeta) \\
& + \mu\eta_x(\xi_y v u_\xi + \xi_x v v_\xi + \xi_z w u_\xi + \xi_x w w_\xi + \zeta_y v u_\zeta + \zeta_x v v_\zeta + \zeta_z w u_\zeta + \zeta_x w w_\zeta) \\
& + \mu\eta_y(\xi_y u u_\xi + \xi_x u v_\xi + \xi_z w v_\xi + \xi_y w w_\xi + \zeta_y u u_\zeta + \zeta_x u v_\zeta + \zeta_z w v_\zeta + \zeta_y w w_\zeta) \\
& + \mu\eta_z(\xi_z u u_\xi + \xi_x u w_\xi + \xi_z v v_\xi + \xi_y v w_\xi + \zeta_z u u_\zeta + \zeta_x u w_\zeta + \zeta_z v v_\zeta + \zeta_y v w_\zeta) \\
& + \frac{k}{Pr_r}(\eta_x\xi_x + \eta_y\xi_y + \eta_z\xi_z)T_\xi + \frac{k}{Pr_r}(\eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z)T_\zeta \Big\}
\end{aligned}
$$

$$
\begin{aligned}
(\hat{\mathbf{G}}_{V_2})_5 = \frac{1}{J}\frac{1}{Re_r}\Big\{ & 2\mu[\zeta_x(\eta_x u u_\eta + \xi_x u u_\xi) + \zeta_y(\eta_y v v_\eta + \xi_y v v_\xi) + \zeta_z(\eta_z w w_\eta + \xi_z w w_\xi)] \\
& + \lambda\zeta_x(\eta_x u u_\eta + \eta_y u v_\eta + \eta_z u w_\eta + \xi_x u u_\xi + \xi_y u v_\xi + \xi_z u w_\xi) \\
& + \lambda\zeta_y(\eta_x v u_\eta + \eta_y v v_\eta + \eta_z v w_\eta + \xi_x v u_\xi + \xi_y v v_\xi + \xi_z v w_\xi) \\
& + \lambda\zeta_z(\eta_x w u_\eta + \eta_y w v_\eta + \eta_z w w_\eta + \xi_x w u_\xi + \xi_y w v_\xi + \xi_z w w_\xi) \\
& + \mu\zeta_x(\eta_y v u_\eta + \eta_x v v_\eta + \eta_z w u_\eta + \eta_x w w_\eta + \xi_y v u_\xi + \xi_x v v_\xi + \xi_z w u_\xi + \xi_x w w_\xi) \\
& + \mu\zeta_y(\eta_y u u_\eta + \eta_x u v_\eta + \eta_z w v_\eta + \eta_y w w_\eta + \xi_y u u_\xi + \xi_x u v_\xi + \xi_z w v_\xi + \xi_y w w_\xi) \\
& + \mu\zeta_z(\eta_z u u_\eta + \eta_x u w_\eta + \eta_z v v_\eta + \eta_y v w_\eta + \xi_z u u_\xi + \xi_x u w_\xi + \xi_z v v_\xi + \xi_y v w_\xi) \\
& + \frac{k}{Pr_r}(\zeta_x\eta_x + \zeta_y\eta_y + \zeta_z\eta_z)T_\eta + \frac{k}{Pr_r}(\zeta_x\xi_x + \zeta_y\xi_y + \zeta_z\xi_z)T_\xi \Big\}
\end{aligned}
$$

The cross-derivative part of the first-sweep source term is computed first. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1. This is followed by the coding for the source term for the second and third sweeps, which consists only of $\Delta\hat{E}_T^*$ or $\Delta\hat{E}_T^{**}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variable S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on S corresponds to the equation.
3. The Euler option is implemented simply by skipping the calculation of the source terms for the viscous and heat conduction terms.
4. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous and heat conduction terms containing derivatives in the specified direction.

Subroutine COEFX		
Called by	Calls	Purpose
EXEC		Compute coefficients and source term for the x-momentum equation.

Input

	DEL	Computational grid spacing in sweep direction.
	DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial\rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
	DTAU	Time step $\Delta\tau$.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
	IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
*	IEULER	Flag for Euler calculation.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	ISWEEP	Current ADI sweep number.
*	ITHIN	Flags for thin-layer option.
	IV	Index in the "vectorized" direction, i_v.
	I2, I3	Grid indices j and k, in the η and ζ directions.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
	MU, LA	Effective coefficient of viscosity μ and effective second coefficient of viscosity λ at time level n.
	NPTS	Number of grid points in the sweep direction, N.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
	NXM	Array index associated with the x-momentum equation.
	P	Static pressure p at time level n.
*	RER	Reference Reynolds number Re_r.
	RHO, U, V, W	Static density ρ, and velocities u, v, and w at time level n.
	RHOL, UL, VL, WL	Static density ρ, and velocities u, v, and w from previous ADI sweep.
*	THX	Parameters θ_1, θ_2, and θ_3 determining type of time differencing for the x-momentum equation.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
	ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** at interior points (row NXM only).
S	Source term subvector **S** at interior points (element NXM only).

Description

Subroutine COEFX computes the coefficients and source term for the x-momentum equation. Equations (7.5a-c) in Volume 1 represent, in vector form, the five governing difference equations for the three ADI sweeps. The elements of the inviscid flux vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{\mathbf{E}}_{V_1}$, $\hat{\mathbf{E}}_{V_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$, $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the x-momentum equation may be written for the three ADI sweeps as[20]

Sweep 1 (ξ direction)

$$
\begin{aligned}
\Delta(\hat{\rho u})_i^* &+ \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\xi}\left[\left(\frac{\partial\hat{\mathbf{E}}_2}{\partial\hat{\mathbf{Q}}}\right)^n_{i+1}\Delta\hat{\mathbf{Q}}^*_{i+1} - \left(\frac{\partial\hat{\mathbf{E}}_2}{\partial\hat{\mathbf{Q}}}\right)^n_{i-1}\Delta\hat{\mathbf{Q}}^*_{i-1}\right] \\
&- \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\xi)^2}\left[(f_{i-1}+f_i)^n g^n_{i-1}\Delta\hat{\mathbf{Q}}^*_{i-1} - (f_{i-1}+2f_i+f_{i+1})^n g^n_i\Delta\hat{\mathbf{Q}}^*_i + (f_i+f_{i+1})^n g^n_{i+1}\Delta\hat{\mathbf{Q}}^*_{i+1}\right] = \\
&- \frac{\Delta\tau}{1+\theta_2}\left(\delta_\xi\hat{\mathbf{E}}_2 + \delta_\eta\hat{\mathbf{F}}_2 + \delta_\zeta\hat{\mathbf{G}}_2\right)^n + \frac{\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_1})_2 + \delta_\eta(\hat{\mathbf{F}}_{V_1})_2 + \delta_\zeta(\hat{\mathbf{G}}_{V_1})_2\right]^n \\
&+ \frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_2 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_2 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_2\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_2 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_2 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_2\right]^{n-1} \\
&+ \frac{\theta_2}{1+\theta_2}\Delta(\hat{\rho u})^{n-1}
\end{aligned}
$$

Sweep 2 (η direction)

$$
\begin{aligned}
\Delta(\hat{\rho u})_j^{**} &+ \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\eta}\left[\left(\frac{\partial\hat{\mathbf{F}}_2}{\partial\hat{\mathbf{Q}}}\right)^n_{j+1}\Delta\hat{\mathbf{Q}}^{**}_{j+1} - \left(\frac{\partial\hat{\mathbf{F}}_2}{\partial\hat{\mathbf{Q}}}\right)^n_{j-1}\Delta\hat{\mathbf{Q}}^{**}_{j-1}\right] \\
&- \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\eta)^2}\left[(f_{j-1}+f_j)^n g^n_{j-1}\Delta\hat{\mathbf{Q}}^{**}_{j-1} - (f_{j-1}+2f_j+f_{j+1})^n g^n_j\Delta\hat{\mathbf{Q}}^{**}_j + (f_j+f_{j+1})^n g^n_{j+1}\Delta\hat{\mathbf{Q}}^{**}_{j+1}\right] = \\
&\Delta(\hat{\rho u})^*
\end{aligned}
$$

---

[20] These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy E_T would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.

Sweep 3 (ζ direction)

$$\Delta(\hat{\rho u})_k^n + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\zeta}\left[\left(\frac{\partial\hat{\mathbf{G}}_2}{\partial\hat{\mathbf{Q}}}\right)_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n - \left(\frac{\partial\hat{\mathbf{G}}_2}{\partial\hat{\mathbf{Q}}}\right)_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n\right]$$

$$- \frac{\theta_1 \Delta\tau}{(1+\theta_2)2(\Delta\zeta)^2}\left[(f_{k-1}+f_k)^n g_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n - (f_{k-1}+2f_k+f_{j+1})^n g_k^n \Delta\hat{\mathbf{Q}}_k^n + (f_k+f_{k+1})^n g_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n\right] =$$

$$\Delta(\hat{\rho u})^{**}$$

In the above equations, the subscripts i, j, and k represent grid point indices in the ξ, η, and ζ directions. For notational convenience, terms without an explicitly written i, j, or k subscript are understood to be at i, j, or k. On the left hand side, f is the coefficient of $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$, depending on the sweep) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix. Similarly, g is the term in the parentheses following $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{\mathbf{Q}} = \frac{1}{J}\left[\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T\right]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{\mathbf{E}}_2 = \frac{1}{J}\left[(\rho u^2 + p)\xi_x + \rho uv\xi_y + \rho uw\xi_z + \rho u\xi_t\right]$$

$$\hat{\mathbf{F}}_2 = \frac{1}{J}\left[(\rho u^2 + p)\eta_x + \rho uv\eta_y + \rho uw\eta_z + \rho u\eta_t\right]$$

$$\hat{\mathbf{G}}_2 = \frac{1}{J}\left[(\rho u^2 + p)\zeta_x + \rho uv\zeta_y + \rho uw\zeta_z + \rho u\zeta_t\right]$$

The appropriate elements of the non-cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_1})_2 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\xi_x^2 u_\xi + \lambda\xi_x(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi) + \mu\xi_y(\xi_y u_\xi + \xi_x v_\xi) + \mu\xi_z(\xi_z u_\xi + \xi_x w_\xi)\right]$$

$$(\hat{\mathbf{F}}_{V_1})_2 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\eta_x^2 u_\eta + \lambda\eta_x(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta) + \mu\eta_y(\eta_y u_\eta + \eta_x v_\eta) + \mu\eta_z(\eta_z u_\eta + \eta_x w_\eta)\right]$$

$$(\hat{\mathbf{G}}_{V_1})_2 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\zeta_x^2 u_\zeta + \lambda\zeta_x(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta) + \mu\zeta_y(\zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\zeta_z(\zeta_z u_\zeta + \zeta_x w_\zeta)\right]$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_2})_2 = \frac{1}{J}\frac{1}{Re_r}\Big[2\mu\xi_x(\eta_x u_\eta + \zeta_x u_\zeta) + \lambda\xi_x(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\xi_y(\eta_y u_\eta + \eta_x v_\eta + \zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\xi_z(\eta_z u_\eta + \eta_x w_\eta + \zeta_z u_\zeta + \zeta_x w_\zeta)\Big]$$

$$(\hat{\mathbf{F}}_{V_2})_2 = \frac{1}{J}\frac{1}{Re_r}\Big[2\mu\eta_x(\xi_x u_\xi + \zeta_x u_\zeta) + \lambda\eta_x(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\eta_y(\xi_y u_\xi + \xi_x v_\xi + \zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\eta_z(\xi_z u_\xi + \xi_x w_\xi + \zeta_z u_\zeta + \zeta_x w_\zeta)\Big]$$

$$(\hat{\mathbf{G}}_{V_2})_2 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\zeta_x(\eta_x u_\eta + \xi_x u_\xi) + \lambda\zeta_x(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi)\right.$$
$$\left. + \mu\zeta_y(\eta_y u_\eta + \eta_x v_\eta + \xi_y u_\xi + \xi_x v_\xi) + \mu\zeta_z(\eta_z u_\eta + \eta_x w_\eta + \xi_z u_\xi + \xi_x w_\xi)\right]$$

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$ for the inviscid terms in the x-momentum equation are

$$\frac{\partial\hat{\mathbf{E}}_2}{\partial\hat{\mathbf{Q}}} = \left[\; \frac{\partial p}{\partial\rho}\xi_x - uf_1 \quad \xi_t + f_1 + u\xi_x + \frac{\partial p}{\partial(\rho u)}\xi_x \quad u\xi_y + \frac{\partial p}{\partial(\rho v)}\xi_x \quad u\xi_z + \frac{\partial p}{\partial(\rho w)}\xi_x \quad \frac{\partial p}{\partial E_T}\xi_x \;\right]$$

where $f_1 = u\xi_x + v\xi_y + w\xi_z$.

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ for the viscous terms are

$$\frac{\partial(\hat{\mathbf{E}}_{V_1})_2}{\partial\hat{\mathbf{Q}}} = \frac{1}{Re_r}\left[\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{21} \quad \alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) \quad \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) \quad \alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) \quad 0\right]$$

where

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{21} = -\alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) - \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) - \alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right)$$

$$\alpha_{xx} = (2\mu + \lambda){\xi_x}^2 + \mu{\xi_y}^2 + \mu{\xi_z}^2$$
$$\alpha_{xy} = (\mu + \lambda)\xi_x\xi_y$$
$$\alpha_{xz} = (\mu + \lambda)\xi_x\xi_z$$

The Jacobian coefficient matrices $\partial\hat{\mathbf{F}}_2/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{F}}_{V_1})_2/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_2/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_2/\partial\hat{\mathbf{Q}}$, but with ξ replaced by η. Similarly, the Jacobian coefficient matrices $\partial\hat{\mathbf{G}}_2/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{G}}_{V_1})_2/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_2/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_2/\partial\hat{\mathbf{Q}}$, but with ξ replaced by ζ.

As an example of how these equations are translated into Fortran, consider the $\Delta(\rho u/J)$ term on the left hand side for the first sweep. This is the second element of $\hat{\mathbf{Q}}$, so using the second element in $\partial\hat{\mathbf{E}}_2/\partial\hat{\mathbf{Q}}$, and including the $\Delta(\hat{\rho u})_i^*$ term, we get for the inviscid term

$$\text{A(IV,I,NXM,NRU)} = -\frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(\xi_t)_{i-1,j,k} + (u\xi_x + v\xi_y + w\xi_z)_{i-1,j,k} + (u\xi_x)_{i-1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_x\right)_{i-1,j,k}\right]$$

$$\text{B(IV,I,NXM,NRU)} = 1$$

$$\text{C(IV,I,NXM,NRU)} = \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(\xi_t)_{i+1,j,k} + (u\xi_x + v\xi_y + w\xi_z)_{i+1,j,k} + (u\xi_x)_{i+1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_x\right)_{i+1,j,k}\right]$$

For the viscous terms on the left hand side, we use the second element in $\partial(\hat{\mathbf{E}}_{V_1})_2/\partial\hat{\mathbf{Q}}$, which is

$$\frac{1}{Re_r}\alpha_{xx}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right)$$

Thus $f = \alpha_{xx}/Re_r$ and $g = 1/\rho$. To add the viscous contribution to this part of the A coefficient submatrix, we therefore set

$$\mathrm{A(IV,I,NXM,NRU)} = \mathrm{A(IV,I,NXM,NRU)} - \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2(\Delta\xi)^2 Re_r}\left[(\alpha_{xx})_{i-1,j,k} + (\alpha_{xx})_{i,j,k}\right]\left(\frac{1}{\rho}\right)_{i-1,j,k}$$

Similar equations may be written for the **B** and **C** coefficient submatrices.

In COEFX, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second and third ADI sweeps have exactly the same form as for the first sweep, but with ξ replaced by η and ζ, respectively. By defining DEL, METX, METY, METZ, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for all three sweeps.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1. This is followed by the coding for the source term for the second and third sweeps, which consists only of $\Delta(\hat{\rho u})^*$ or $\Delta(\hat{\rho u})^{**}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply to all three sweeps (i.e., the implicit terms), the first two subscripts are written as (IV,I). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.
3. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.
4. The thin-layer option is implemented by skipping the calculation of the coefficients and source erms for the viscous terms containing derivatives in the specified direction.

Subroutine COEFY		
Called by	Calls	Purpose
EXEC		Compute coefficients and source term for the y-momentum equation.

Input

	DEL	Computational grid spacing in sweep direction.
	DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial\rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
	DTAU	Time step $\Delta\tau$.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
	IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
*	IEULER	Flag for Euler calculation.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	ISWEEP	Current ADI sweep number.
*	ITHIN	Flags for thin-layer option.
	IV	Index in the "vectorized" direction, i_v.
	I2, I3	Grid indices j and k, in the η and ζ directions.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
	MU, LA	Effective coefficient of viscosity μ and effective second coefficient of viscosity λ at time level n.
	NPTS	Number of grid points in the sweep direction, N.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
	NYM	Array index associated with the y-momentum equation.
	P	Static pressure p at time level n.
*	RER	Reference Reynolds number Re_r.
	RHO, U, V, W	Static density ρ, and velocities u, v, and w at time level n.
	RHOL, UL, VL, WL	Static density ρ, and velocities u, v, and w from previous ADI sweep.
*	THY	Parameters θ_1, θ_2, and θ_3 determining type of time differencing for the y-momentum equation.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
	ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

A, B, C — Coefficient submatrices **A**, **B**, and **C** at interior points (row NYM only).

S — Source term subvector **S** at interior points (element NYM only).

Description

Subroutine COEFY computes the coefficients and source term for the y-momentum equation. Equations (7.5a-c) in Volume 1 represent, in vector form, the five governing difference equations for the three ADI sweeps. The elements of the inviscid flux vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{\mathbf{E}}_{V_1}$, $\hat{\mathbf{E}}_{V_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$, $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the y-momentum equation may be written for the three ADI sweeps as[21]

Sweep 1 (ξ direction)

$$
\begin{aligned}
&\Delta(\hat{\rho v})_i^* + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\xi}\left[\left(\frac{\partial\hat{\mathbf{E}}_3}{\partial\hat{\mathbf{Q}}}\right)_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^* - \left(\frac{\partial\hat{\mathbf{E}}_3}{\partial\hat{\mathbf{Q}}}\right)_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^*\right] \\
&\quad - \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\xi)^2}\left[(f_{i-1}+f_i)^n g_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^* - (f_{i-1}+2f_i+f_{i+1})^n g_i^n \Delta\hat{\mathbf{Q}}_i^* + (f_i+f_{i+1})^n g_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^*\right] = \\
&\quad - \frac{\Delta\tau}{1+\theta_2}\left(\delta_\xi\hat{\mathbf{E}}_3 + \delta_\eta\hat{\mathbf{F}}_3 + \delta_\zeta\hat{\mathbf{G}}_3\right)^n + \frac{\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_1})_3 + \delta_\eta(\hat{\mathbf{F}}_{V_1})_3 + \delta_\zeta(\hat{\mathbf{G}}_{V_1})_3\right]^n \\
&\quad + \frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_3 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_3 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_3\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_3 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_3 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_3\right]^{n-1} \\
&\quad + \frac{\theta_2}{1+\theta_2}\Delta(\hat{\rho v})^{n-1}
\end{aligned}
$$

Sweep 2 (η direction)

$$
\begin{aligned}
&\Delta(\hat{\rho v})_j^{**} + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\eta}\left[\left(\frac{\partial\hat{\mathbf{F}}_3}{\partial\hat{\mathbf{Q}}}\right)_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**} - \left(\frac{\partial\hat{\mathbf{F}}_3}{\partial\hat{\mathbf{Q}}}\right)_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**}\right] \\
&\quad - \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\eta)^2}\left[(f_{j-1}+f_j)^n g_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**} - (f_{j-1}+2f_j+f_{j+1})^n g_j^n \Delta\hat{\mathbf{Q}}_j^{**} + (f_j+f_{j+1})^n g_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**}\right] = \\
&\quad \Delta(\hat{\rho v})^*
\end{aligned}
$$

---

[21] These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy E_T would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.

Sweep 3 (ζ direction)

$$\Delta(\hat{\rho v})_j^n + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\zeta}\left[\left(\frac{\partial\hat{\mathbf{G}}_3}{\partial\hat{\mathbf{Q}}}\right)_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n - \left(\frac{\partial\hat{\mathbf{G}}_3}{\partial\hat{\mathbf{Q}}}\right)_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n\right]$$

$$-\frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\zeta)^2}\left[(f_{k-1}+f_k)^n g_{k-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^n - (f_{k-1}+2f_k+f_{k+1})^n g_k^n \Delta\hat{\mathbf{Q}}_j^n + (f_k+f_{k+1})^n g_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n\right] =$$
$$\Delta(\hat{\rho v})^{**}$$

In the above equations, the subscripts i, j, and k represent grid point indices in the ξ, η, and ζ directions. For notational convenience, terms without an explicitly written i, j, or k subscript are understood to be at i, j, or k. On the left hand side, f is the coefficient of $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$, depending on the sweep) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix. Similarly, g is the term in the parentheses following $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{\mathbf{Q}} = \frac{1}{J}[\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{\mathbf{E}}_3 = \frac{1}{J}[\rho uv\xi_x + (\rho v^2 + p)\xi_y + \rho vw\xi_z + \rho v\xi_t]$$

$$\hat{\mathbf{F}}_3 = \frac{1}{J}[\rho uv\eta_x + (\rho v^2 + p)\eta_y + \rho vw\eta_z + \rho v\eta_t]$$

$$\hat{\mathbf{G}}_3 = \frac{1}{J}[\rho uv\zeta_x + (\rho v^2 + p)\zeta_y + \rho vw\zeta_z + \rho v\zeta_t]$$

The appropriate elements of the non-cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_1})_3 = \frac{1}{J}\frac{1}{Re_r}[2\mu\xi_y^2 v_\xi + \lambda\xi_y(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi) + \mu\xi_x(\xi_y u_\xi + \xi_x v_\xi) + \mu\xi_z(\xi_z v_\xi + \xi_y w_\xi)]$$

$$(\hat{\mathbf{F}}_{V_1})_3 = \frac{1}{J}\frac{1}{Re_r}[2\mu\eta_y^2 v_\eta + \lambda\eta_y(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta) + \mu\eta_x(\eta_y u_\eta + \eta_x v_\eta) + \mu\eta_z(\eta_z v_\eta + \eta_y w_\eta)]$$

$$(\hat{\mathbf{G}}_{V_1})_3 = \frac{1}{J}\frac{1}{Re_r}[2\mu\zeta_y^2 v_\zeta + \lambda\zeta_y(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta) + \mu\zeta_x(\zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\zeta_z(\zeta_z v_\zeta + \zeta_y w_\zeta)]$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_2})_3 = \frac{1}{J}\frac{1}{Re_r}\big[2\mu\xi_y(\eta_y v_\eta + \zeta_y v_\zeta) + \lambda\xi_y(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\xi_x(\eta_y u_\eta + \eta_x v_\eta + \zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\xi_z(\eta_z v_\eta + \eta_y w_\eta + \zeta_z v_\zeta + \zeta_y w_\zeta)\big]$$

$$(\hat{\mathbf{F}}_{V_2})_3 = \frac{1}{J}\frac{1}{Re_r}\big[2\mu\eta_y(\xi_y v_\xi + \zeta_y v_\zeta) + \lambda\eta_y(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\eta_x(\xi_y u_\xi + \xi_x v_\xi + \zeta_y u_\zeta + \zeta_x v_\zeta) + \mu\eta_z(\xi_z v_\xi + \xi_y w_\xi + \zeta_z v_\zeta + \zeta_y w_\zeta)\big]$$

$$(\hat{\mathbf{G}}_{V_2})_3 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\zeta_y(\eta_y v_\eta + \xi_y v_\xi) + \lambda\zeta_y(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi)\right.$$
$$\left. + \mu\zeta_x(\eta_y u_\eta + \eta_x v_\eta + \xi_y u_\xi + \xi_x v_\xi) + \mu\zeta_z(\eta_z v_\eta + \eta_y w_\eta + \xi_z v_\xi + \xi_y w_\xi)\right]$$

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$ for the inviscid terms in the y-momentum equation are

$$\frac{\partial\hat{\mathbf{E}}_3}{\partial\hat{\mathbf{Q}}} = \left[\begin{array}{ccccc} \frac{\partial p}{\partial\rho}\xi_y - vf_1 & v\xi_x + \frac{\partial p}{\partial(\rho u)}\xi_y & \xi_t + f_1 + v\xi_y + \frac{\partial p}{\partial(\rho v)}\xi_y & v\xi_z + \frac{\partial p}{\partial(\rho w)}\xi_y & \frac{\partial p}{\partial E_T}\xi_y \end{array}\right]$$

where $f_1 = u\xi_x + v\xi_y + w\xi_z$.

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ for the viscous terms are

$$\frac{\partial(\hat{\mathbf{E}}_{V_1})_3}{\partial\hat{\mathbf{Q}}} = \frac{1}{Re_r}\left[\begin{array}{ccccc} \left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{31} & \alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) & \alpha_{yy}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) & \alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right) & 0 \end{array}\right]$$

where

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{31} = -\alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) - \alpha_{yy}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) - \alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right)$$

$$\alpha_{xy} = (\mu + \lambda)\xi_x\xi_y$$

$$\alpha_{yy} = \mu\xi_x^{\,2} + (2\mu + \lambda)\xi_y^{\,2} + \mu\xi_z^{\,2}$$

$$\alpha_{yz} = (\mu + \lambda)\xi_y\xi_z$$

The Jacobian coefficient matrices $\partial\hat{\mathbf{F}}_3/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{F}}_{V_1})_3/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_3/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_3/\partial\hat{\mathbf{Q}}$, but with ξ replaced by η. Similarly, the Jacobian coefficient matrices $\partial\hat{\mathbf{G}}_3/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{G}}_{V_1})_3/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_3/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_3/\partial\hat{\mathbf{Q}}$, but with ξ replaced by ζ.

As an example of how these equations are translated into Fortran, consider the $\Delta(\rho u/J)$ term on the left hand side for the first sweep. This is the second element of $\hat{\mathbf{Q}}$, so using the second element in $\partial\hat{\mathbf{E}}_3/\partial\hat{\mathbf{Q}}$, we get for the inviscid term

$$\text{A(IV,I,NYM,NRU)} = -\frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(v\xi_x)_{i-1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_y\right)_{i-1,j,k}\right]$$

$$\text{B(IV,I,NYM,NRU)} = 0$$

$$\text{C(IV,I,NYM,NRU)} = \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(v\xi_x)_{i+1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_y\right)_{i+1,j,k}\right]$$

For the viscous terms on the left hand side, we use the second element in $\partial(\hat{\mathbf{E}}_{V_1})_3/\partial\hat{\mathbf{Q}}$, which is

$$\frac{1}{Re_r}\alpha_{xy}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right)$$

Thus $f = \alpha_{xy}/Re_r$ and $g = 1/\rho$. To add the viscous contribution to this part of the A coefficient submatrix, we therefore set

$$\text{A(IV,I,NYM,NRU)} = \text{A(IV,I,NYM,NRU)} - \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2(\Delta\xi)^2 Re_r}\left[(\alpha_{xy})_{i-1,j,k} + (\alpha_{xy})_{i,j,k}\right]\left(\frac{1}{\rho}\right)_{i-1,j,k}$$

Similar equations may be written for the **B** and **C** coefficient submatrices.

In COEFY, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second and third ADI sweeps have exactly the same form as for the first sweep, but with ξ replaced by η and ζ, respectively. By defining DEL, METX, METY, METZ, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for all three sweeps.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1. This is followed by the coding for the source term for the second and third sweeps, which consists only of $\Delta(\hat{\rho v})^*$ or $\Delta(\hat{\rho v})^{**}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply to all three sweeps (i.e., the implicit terms), the first two subscripts are written as (IV,I). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.
3. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.
4. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous terms containing derivatives in the specified direction.

Subroutine COEFZ		
Called by	Calls	Purpose
EXEC		Compute coefficients and source term for the z-momentum equation.

Input

	DEL	Computational grid spacing in sweep direction.
	DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial \rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
	DTAU	Time step $\Delta\tau$.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
	IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
*	IEULER	Flag for Euler calculation.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	ISWEEP	Current ADI sweep number.
*	ITHIN	Flags for thin-layer option.
	IV	Index in the "vectorized" direction, i_v.
	I2, I3	Grid indices j and k, in the η and ζ directions.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
	MU, LA	Effective coefficient of viscosity μ and effective second coefficient of viscosity λ at time level n.
	NPTS	Number of grid points in the sweep direction, N.
	NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
	NZM	Array index associated with the z-momentum equation.
	P	Static pressure p at time level n.
*	RER	Reference Reynolds number Re_r.
	RHO, U, V, W	Static density ρ, and velocities u, v, and w at time level n.
	RHOL, UL, VL, WL	Static density ρ, and velocities u, v, and w from previous ADI sweep.
*	THZ	Parameters θ_1, θ_2, and θ_3 determining type of time differencing for the z-momentum equation.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
	ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

A, B, C — Coefficient submatrices **A**, **B**, and **C** at interior points (row NZM only).

S — Source term subvector **S** at interior points (element NZM only).

Description

Subroutine COEFZ computes the coefficients and source term for the z-momentum equation. Equations (7.5a-c) in Volume 1 represent, in vector form, the five governing difference equations for the three ADI sweeps. The elements of the inviscid flux vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{\mathbf{E}}_{V_1}$, $\hat{\mathbf{E}}_{V_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$, $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the z-momentum equation may be written for the three ADI sweeps as[22]

Sweep 1 (ξ direction)

$$
\begin{aligned}
&\Delta(\rho\hat{w})_i^* + \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\xi}\left[\left(\frac{\partial\hat{\mathbf{E}}_4}{\partial\hat{\mathbf{Q}}}\right)_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^* - \left(\frac{\partial\hat{\mathbf{E}}_4}{\partial\hat{\mathbf{Q}}}\right)_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^*\right] \\
&\quad - \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\xi)^2}\left[(f_{i-1}+f_i)^n g_{i-1}^n \Delta\hat{\mathbf{Q}}_{i-1}^* - (f_{i-1}+2f_i+f_{i+1})^n g_i^n \Delta\hat{\mathbf{Q}}_i^* + (f_i+f_{i+1})^n g_{i+1}^n \Delta\hat{\mathbf{Q}}_{i+1}^*\right] = \\
&\quad - \frac{\Delta\tau}{1+\theta_2}\left(\delta_\xi\hat{\mathbf{E}}_4 + \delta_\eta\hat{\mathbf{F}}_4 + \delta_\zeta\hat{\mathbf{G}}_4\right)^n + \frac{\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_1})_4 + \delta_\eta(\hat{\mathbf{F}}_{V_1})_4 + \delta_\zeta(\hat{\mathbf{G}}_{V_1})_4\right]^n \\
&\quad + \frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_4 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_4 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_4\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\delta_\xi(\hat{\mathbf{E}}_{V_2})_4 + \delta_\eta(\hat{\mathbf{F}}_{V_2})_4 + \delta_\zeta(\hat{\mathbf{G}}_{V_2})_4\right]^{n-1} \\
&\quad + \frac{\theta_2}{1+\theta_2}\Delta(\rho\hat{w})^{n-1}
\end{aligned}
$$

Sweep 2 (η direction)

$$
\begin{aligned}
&\Delta(\rho\hat{w})_j^{**} + \frac{\theta_1\Delta\tau}{(1+\theta_2)2\Delta\eta}\left[\left(\frac{\partial\hat{\mathbf{F}}_4}{\partial\hat{\mathbf{Q}}}\right)_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**} - \left(\frac{\partial\hat{\mathbf{F}}_4}{\partial\hat{\mathbf{Q}}}\right)_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**}\right] \\
&\quad - \frac{\theta_1\Delta\tau}{(1+\theta_2)2(\Delta\eta)^2}\left[(f_{j-1}+f_j)^n g_{j-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^{**} - (f_{j-1}+2f_j+f_{j+1})^n g_j^n \Delta\hat{\mathbf{Q}}_j^{**} + (f_j+f_{j+1})^n g_{j+1}^n \Delta\hat{\mathbf{Q}}_{j+1}^{**}\right] = \\
&\quad \Delta(\rho\hat{w})^*
\end{aligned}
$$

---

[22] These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy E_T would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.

Sweep 3 (ζ direction)

$$\Delta(\hat{\rho w})_j^n + \frac{\theta_1 \Delta\tau}{(1+\theta_2)2\Delta\zeta}\left[\left(\frac{\partial\hat{\mathbf{G}}_4}{\partial\hat{\mathbf{Q}}}\right)_{k-1}^n \Delta\hat{\mathbf{Q}}_{k-1}^n - \left(\frac{\partial\hat{\mathbf{G}}_4}{\partial\hat{\mathbf{Q}}}\right)_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n\right]$$

$$- \frac{\theta_1 \Delta\tau}{(1+\theta_2)2(\Delta\zeta)^2}\left[(f_{k-1}+f_k)^n g_{k-1}^n \Delta\hat{\mathbf{Q}}_{j-1}^n - (f_{k-1}+2f_k+f_{k+1})^n g_k^n \Delta\hat{\mathbf{Q}}_j^n + (f_k+f_{k+1})^n g_{k+1}^n \Delta\hat{\mathbf{Q}}_{k+1}^n\right] =$$

$$\Delta(\hat{\rho w})^{**}$$

In the above equations, the subscripts i, j, and k represent grid point indices in the ξ, η, and ζ directions. For notational convenience, terms without an explicitly written i, j, or k subscript are understood to be at i, j, or k. On the left hand side, f is the coefficient of $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$, depending on the sweep) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix. Similarly, g is the term in the parentheses following $\partial/\partial\xi$ (or $\partial/\partial\eta$ or $\partial/\partial\zeta$) in the $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ (or $\partial\hat{\mathbf{F}}_{V_1}/\partial\hat{\mathbf{Q}}$ or $\partial\hat{\mathbf{G}}_{V_1}/\partial\hat{\mathbf{Q}}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{\mathbf{Q}} = \frac{1}{J}\left[\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T\right]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{\mathbf{E}}_4 = \frac{1}{J}\left[\rho u w \xi_x + \rho v w \xi_y + (\rho w^2 + p)\xi_z + \rho w \xi_t\right]$$

$$\hat{\mathbf{F}}_4 = \frac{1}{J}\left[\rho u w \eta_x + \rho v w \eta_y + (\rho w^2 + p)\eta_z + \rho w \eta_t\right]$$

$$\hat{\mathbf{G}}_4 = \frac{1}{J}\left[\rho u w \zeta_x + \rho v w \zeta_y + (\rho w^2 + p)\zeta_z + \rho w \zeta_t\right]$$

The appropriate elements of the non-cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_1})_4 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\xi_z^2 w_\xi + \lambda\xi_z(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi) + \mu\xi_x(\xi_z u_\xi + \xi_x w_\xi) + \mu\xi_y(\xi_z v_\xi + \xi_y w_\xi)\right]$$

$$(\hat{\mathbf{F}}_{V_1})_4 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\eta_z^2 w_\eta + \lambda\eta_z(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta) + \mu\eta_x(\eta_z u_\eta + \eta_x w_\eta) + \mu\eta_y(\eta_z v_\eta + \eta_y w_\eta)\right]$$

$$(\hat{\mathbf{G}}_{V_1})_4 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\zeta_z^2 w_\zeta + \lambda\zeta_z(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta) + \mu\zeta_x(\zeta_z u_\zeta + \zeta_x w_\zeta) + \mu\zeta_y(\zeta_z v_\zeta + \zeta_y w_\zeta)\right]$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$(\hat{\mathbf{E}}_{V_2})_4 = \frac{1}{J}\frac{1}{Re_r}\big[2\mu\xi_z(\eta_z w_\eta + \zeta_z w_\zeta) + \lambda\xi_z(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\xi_x(\eta_z u_\eta + \eta_x w_\eta + \zeta_z u_\zeta + \zeta_x w_\zeta) + \mu\xi_y(\eta_z v_\eta + \eta_y w_\eta + \zeta_z v_\zeta + \zeta_y w_\zeta)\big]$$

$$(\hat{\mathbf{F}}_{V_2})_4 = \frac{1}{J}\frac{1}{Re_r}\big[2\mu\eta_z(\xi_z w_\xi + \zeta_z w_\zeta) + \lambda\eta_z(\xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi + \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)$$
$$+ \mu\eta_x(\xi_z u_\xi + \xi_x w_\xi + \zeta_z u_\zeta + \zeta_x w_\zeta) + \mu\eta_y(\xi_z v_\xi + \xi_y w_\xi + \zeta_z v_\zeta + \zeta_y w_\zeta)\big]$$

$$(\hat{\mathbf{G}}_{V_2})_4 = \frac{1}{J}\frac{1}{Re_r}\left[2\mu\zeta_z(\eta_z w_\eta + \xi_z w_\xi) + \lambda\zeta_z(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta + \xi_x u_\xi + \xi_y v_\xi + \xi_z w_\xi)\right.$$
$$\left. + \mu\zeta_x(\eta_z u_\eta + \eta_x w_\eta + \xi_z u_\xi + \xi_x w_\xi) + \mu\zeta_y(\eta_z v_\eta + \eta_y w_\eta + \xi_z v_\xi + \xi_y w_\xi)\right]$$

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}/\partial\hat{\mathbf{Q}}$ for the inviscid terms in the z-momentum equation are

$$\frac{\partial\hat{\mathbf{E}}_4}{\partial\hat{\mathbf{Q}}} = \left[\begin{array}{ccccc} \dfrac{\partial p}{\partial\rho}\xi_z - wf_1 & w\xi_x + \dfrac{\partial p}{\partial(\rho u)}\xi_z & w\xi_y + \dfrac{\partial p}{\partial(\rho v)}\xi_z & \xi_t + f_1 + w\xi_z + \dfrac{\partial p}{\partial(\rho w)}\xi_z & \dfrac{\partial p}{\partial E_T}\xi_z \end{array}\right]$$

where $f_1 = u\xi_x + v\xi_y + w\xi_z$.

The elements of the Jacobian coefficient matrix $\partial\hat{\mathbf{E}}_{V_1}/\partial\hat{\mathbf{Q}}$ for the viscous terms are

$$\frac{\partial(\hat{\mathbf{E}}_{V_1})_4}{\partial\hat{\mathbf{Q}}} = \frac{1}{Re_r}\left[\begin{array}{ccccc} \left(\dfrac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{41} & \alpha_{xz}\dfrac{\partial}{\partial\xi}\left(\dfrac{1}{\rho}\right) & \alpha_{yz}\dfrac{\partial}{\partial\xi}\left(\dfrac{1}{\rho}\right) & \alpha_{zz}\dfrac{\partial}{\partial\xi}\left(\dfrac{1}{\rho}\right) & 0 \end{array}\right]$$

where

$$\left(\frac{\partial\hat{\mathbf{E}}_{V_1}}{\partial\hat{\mathbf{Q}}}\right)_{41} = -\alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{u}{\rho}\right) - \alpha_{yz}\frac{\partial}{\partial\xi}\left(\frac{v}{\rho}\right) - \alpha_{zz}\frac{\partial}{\partial\xi}\left(\frac{w}{\rho}\right)$$

$$\alpha_{xz} = (\mu + \lambda)\xi_x\xi_z$$
$$\alpha_{yz} = (\mu + \lambda)\xi_y\xi_z$$
$$\alpha_{zz} = \mu\xi_x^{\,2} + \mu\xi_y^{\,2} + (2\mu + \lambda)\xi_z^{\,2}$$

The Jacobian coefficient matrices $\partial\hat{\mathbf{F}}_4/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{F}}_{V_1})_4/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_4/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_4/\partial\hat{\mathbf{Q}}$, but with ξ replaced by η. Similarly, the Jacobian coefficient matrices $\partial\hat{\mathbf{G}}_4/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{G}}_{V_1})_4/\partial\hat{\mathbf{Q}}$ have the same form as $\partial\hat{\mathbf{E}}_4/\partial\hat{\mathbf{Q}}$ and $\partial(\hat{\mathbf{E}}_{V_1})_4/\partial\hat{\mathbf{Q}}$, but with ξ replaced by ζ.

As an example of how these equations are translated into Fortran, consider the $\Delta(\rho u/J)$ term on the left hand side for the first sweep. This is the second element of $\hat{\mathbf{Q}}$, so using the second element in $\partial\hat{\mathbf{E}}_4/\partial\hat{\mathbf{Q}}$, we get for the inviscid term

$$\text{A(IV,I,NZM,NRU)} = -\frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(w\xi_x)_{i-1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_z\right)_{i-1,j,k}\right]$$

$$\text{B(IV,I,NZM,NRU)} = 0$$

$$\text{C(IV,I,NZM,NRU)} = \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2\Delta\xi}\left[(w\xi_x)_{i+1,j,k} + \left(\frac{\partial p}{\partial(\rho u)}\xi_z\right)_{i+1,j,k}\right]$$

For the viscous terms on the left hand side, we use the second element in $\partial(\hat{\mathbf{E}}_{V_1})_4/\partial\hat{\mathbf{Q}}$, which is

$$\frac{1}{Re_r}\alpha_{xz}\frac{\partial}{\partial\xi}\left(\frac{1}{\rho}\right)$$

Thus $f = \alpha_{xz}/Re_r$ and $g = 1/\rho$. To add the viscous contribution to this part of the A coefficient submatrix, we therefore set

$$A(IV,I,NZM,NRU) = A(IV,I,NZM,NRU) - \frac{\theta_1(\Delta\tau)_{i,j,k}}{(1+\theta_2)2(\Delta\xi)^2 Re_r}[(\alpha_{xz})_{i-1,j,k} + (\alpha_{xz})_{i,j,k}]\left(\frac{1}{\rho}\right)_{i-1,j,k}$$

Similar equations may be written for the **B** and **C** coefficient submatrices.

In COEFZ, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second and third ADI sweeps have exactly the same form as for the first sweep, but with ξ replaced by η and ζ, respectively. By defining DEL, METX, METY, METZ, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for all three sweeps.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms, and are presented in Section 5.0 of Volume 1. This is followed by the coding for the source term for the second and third sweeps, which consists only of $\Delta(\hat{\rho w})^*$ or $\Delta(\hat{\rho w})^{**}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in one of the non-sweep (i.e., "vectorized") directions, and the second subscript is the index in the sweep direction. For the first sweep the order is thus (I2,I1), for the second sweep the order is (I1,I2), and for the third sweep the order is (I1,I3). For sections of the code that apply to all three sweeps (i.e., the implicit terms), the first two subscripts are written as (IV,I). For sections of the code that apply only to the first sweep, the first two subscripts are written as (I2,I1). For sections that apply to the second and third sweeps, they are written as (I1,I). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.
3. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.
4. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous terms containing derivatives in the specified direction.

Subroutine CONV		
Called by	Calls	Purpose
MAIN	ISAMAX	Test computed flow field for convergence.

Input

CHGMAX	Maximum change in absolute value of the dependent variables from time level $n-1$ to n (or over the previous NITAVG − 1 time steps if ICTEST = 2), $\Delta \mathbf{Q}_{max}$.
DUMMY	A three-dimensional scratch array.
* EPS	Convergence level to be reached, ε.
* GAMR	Reference ratio of specific heats, γ_r.
* IAV2E, IAV4E	Flags for second- and fourth-order explicit implicit artificial viscosity.
* ICTEST	Flag for convergence criteria to be used.
* IHSTAG	Flag for constant stagnation enthalpy option.
IT	Current time step number n.
NEQ	Number of coupled equations being solved, N_{eq}.
* NITAVG	Number of time steps in moving average convergence test.
* NOUT	Unit number for standard output.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
NTOTP	Dimensioning parameter specifying the storage required for a full three-dimensional array (i.e., N1P × N2P × N3P).
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
RESAVG	The average absolute value of the residual at time level n, R_{avg}.
RESL2	The L_2 norm of the residual at time level n, R_{L_2}.
RESMAX	The maximum absolute value of the residual at time level n, R_{max}.
RGAS	Gas constant R.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level $n+1$.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.

Output

CHGAVG	Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps, $\Delta \mathbf{Q}_{avg}$.
CHGMAX	Maximum change in absolute value of the dependent variables from time level n to $n+1$ (or over the previous NITAVG time steps if ICTEST = 2), $\Delta \mathbf{Q}_{max}$.
ICONV	Convergence flag; 1 if converged, 0 if not.

Description

Subroutine CONV checks the computed flow field for convergence. Convergence may be based on: (1) the absolute value of the maximum change in the dependent variables over the previous time step; (2) the absolute value of the maximum change in the dependent variables, averaged over the last NITAVG time steps; (3) the L_2 norm of the residual for each equation; (4) the average residual for each equation; or (5) the maximum residual for each equation. These parameters are defined in Section 4.1.6 of Volume 2.

The convergence criteria to be used and the level to be reached are set by the input parameters ICTEST and EPS. Each dependent variable or equation is checked separately, and convergence is declared when the specified level is reached for all of the variables or equations. The same criteria is used for each one, but different levels may be specified.

Subroutine CONV first computes ΔQ_{max}, the absolute value of the maximum change in each dependent variable over all the grid points for the most recent time step. These values are stored in CHGMAX(IVAR,1), where IVAR varies from 1 to NEQ, the number of dependent variables. If ICTEST = 2 (the so-called "moving average" convergence test), CHGMAX(IVAR,2) contains the maximum change for the previous time step, etc.

Then, depending on the value of ICTEST, the chosen convergence criteria is compared with the level to be reached for each dependent variable or equation, and a flag is set if the calculation is converged.

Remarks

1. For ICTEST = 1 or 2, the change in E_T is divided by $\bar{R}/(\gamma_r - 1) + 1/2$. This is equivalent to dividing the dimensional value $\bar{E}_T$ by

$$E_{T_r} = \frac{\rho_r \bar{R} T_r}{\gamma_r - 1} + \frac{\rho_r u_r^2}{2}$$

   This makes the change in total energy the same order of magnitude as the other conservation variables.

2. For ICTEST = 1 or 2, the convergence test is based on (or includes) the change in dependent variables from time level n to $n+1$. For ICTEST = 3, 4, or 5, convergence is based on the residual at time level n, not $n+1$. This is because the residuals at time level $n+1$ are not computed until the marching step from $n+1$ to $n+2$ is taken.
3. For cases run with artificial viscosity, the residuals are computed and printed both with and without the artificial viscosity terms. This may provide some estimate of the overall error in the solution introduced by the artificial viscosity. Convergence is determined by the residuals with the artificial viscosity terms included.
4. The Cray search routine ISAMAX is used in computing the absolute value of the maximum change in dependent variables.
5. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the change in dependent variables for use by ISAMAX.
6. A warning message is generated if an illegal convergence criteria is specified. ICTEST is reset to 3 (convergence based on the L_2 norm of the residual), and the calculation will continue.

Subroutine CUBIC (IDIR,T,G,NOLD,TINT,GINT)		
Called by	Calls	Purpose
PAK		Interpolation using Ferguson's parametric cubic.

Input

G — A three-dimensional array containing NOLD1 × NOLD2 × NOLD3 values of the function $g(t)$ to be interpolated.

IDIR — Direction flag; 1 if first subscript in G varies, 2 if second subscript varies, 3 if third subscript varies.

I1, I2, I3 — Grid indices i, j, and k, in the ξ, η, and ζ directions.

NOLD — Number of values in direction IDIR in array G (i.e., NOLD1, NOLD2, or NOLD3.)

* N1, N2, N3 — Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

T — A one-dimensional array containing NOLD values of the independent variable t.

TINT — A one-dimensional array containing N1, N2, or N3 (depending on IDIR) values of the independent variable $t = t_{int}$ at which interpolated values $g_{int} = g(t_{int})$ are desired.

Output

GINT — A one-dimensional array containing N1, N2, or N3 (depending on IDIR) interpolated values $g_{int} = g(t_{int})$.

Description

Subroutine CUBIC performs interpolation using Ferguson's parametric cubic polynomial (Faux and Pratt, 1979). Given the function $g(t)$ and a value t_{int}, CUBIC computes $g_{int} = g(t_{int})$.

The function $g(t)$ is specified by the Fortran arrays G and T. For a general value t, let

$$t_f = \frac{t - t_u}{t_d - t_u}$$

where $t_u \le t \le t_d$. (I.e., t_u and t_d are the two elements of the array T that bracket t.)

Between t_u and t_d, assume g can be described by a cubic polynomial in t_f, as follows:

$$g = a_1 + a_2 t_f + a_3 t_f^2 + a_4 t_f^3$$

Differentiating,

$$g' = \frac{dg}{dt_f} = a_2 + 2a_3 t_f + 3a_4 t_f^2$$

Noting that $t_f = 0$ at $t = t_u$, and 1 at $t = t_d$, we get

$$g_u = a_1$$
$$g'_u = a_2$$
$$g_d = a_1 + a_2 + a_3 + a_4$$
$$g'_d = a_2 + 2a_3 + 3a_4$$

Solving for a_1 through a_4,

$$a_1 = g_u$$
$$a_2 = g'_u$$
$$a_3 = 3(g_d - g_u) - 2g'_u - g'_d$$
$$a_4 = 2(g_u - g_d) + g'_u + g'_d$$

Plugging these into the cubic polynomial for f and rearranging,

$$g = g_u(1 - 3t_f^2 + 2t_f^3) + g_d(3t_f^2 - 2t_f^3) + g'_u(t_f - 2t_f^2 + t_f^3) + g'_d(-t_f^2 + t_f^3)$$

This is the form of the equation used to compute g_{int}.

Remarks

1. At interior points in the array g, the derivatives g'_u and g'_d are computed using a second-order central difference formula. At the end points, second-order one-sided difference formulas are used.
2. The Fortran variable TINT is actually a one-dimensional array containing N_1, N_2, or N_3 input values of t_{int}. Similarly, GINT is a one-dimensional array containing N_1, N_2, or N_3 output values of g_{int}.
3. The Fortran array G that specifies the input values of $g(t)$ is actually a three-dimensional array. Within CUBIC, however, only one of the subscripts varies. The input flag IDIR specifies which one.

Subroutine EQSTAT (ICALL)		
Called by	Calls	Purpose
BVUP EXEC INITC MAIN		Use equation of state to compute pressure, temperature, and their derivatives with respect to the dependent variables.

Input

CP, CV	Specific heats c_p and c_v.
* HSTAG	Stagnation enthalpy h_T used with constant stagnation enthalpy option.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
ICALL	0 to get p and T, 1 to get derivatives of p and T with respect to dependent variables.
* IHSTAG	Flag for constant stagnation enthalpy option.
NPTS	Number of grid points in the sweep direction, N.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
RGAS	Gas constant R.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T.

Output

DPDRHO, DPDRU, DPDRV, DPDRW, DPDET	Derivatives $\partial p/\partial\rho$, $\partial p/\partial(\rho u)$, $\partial p/\partial(\rho v)$, $\partial p/\partial(\rho w)$, and $\partial p/\partial E_T$.
DTDRHO, DTDRU, DTDRV, DTDRW, DTDET	Derivatives $\partial T/\partial\rho$, $\partial T/\partial(\rho u)$, $\partial T/\partial(\rho v)$, $\partial T/\partial(\rho w)$, and $\partial T/\partial E_T$.
ET	Total energy (constant stagnation enthalpy option only.)
INEG	Flag for non-positive pressure and/or temperature; 0 if positive, 1 if non-positive.
P, T	Static pressure p and temperature T.

Description

Subroutine EQSTAT computes various quantities that depend on the form of the equation of state. It actually serves a dual purpose. First, it is called from subroutine INITC and from the MAIN program, with the input parameter ICALL = 0, to compute the static pressure p and temperature T from the initial or just-computed values of the dependent variables. If the constant stagnation enthalpy option is being used it also computes a value for the total energy E_T. And second, it is called from subroutines BVUP and EXEC, with ICALL = 1, to compute the derivatives of p and T with respect to the dependent variables.[23]

The equation of state currently built into *Proteus* is for a perfect gas. The formulas used to compute p, T, and their derivatives with respect to the dependent variables are presented in Section 4.3 of Volume 1.

[23] These are needed for linearization of the governing equations. See Section 4.1 of Volume 1 for details.

Remarks

1. When used to compute p and T (ICALL = 0), this subroutine is called from outside any loops in the ξ, η, or ζ directions. When used to compute $\partial p/\partial\rho$, etc., (ICALL = 1), it is called for each ADI sweep from inside a loop in the non-sweep direction.
2. When computing $\partial p/\partial\rho$, etc., this subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.

Subroutine EXEC		
Called by	Calls	Purpose
MAIN	ADI AVISC1 AVISC2 BCELIM BCGEN BVUP COEFC COEFE1 COEFE2 COEFX COEFY COEFZ EQSTAT PERIOD RESID UPDATE	Manage solution of governing equations.

Input

	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
*	IAV2E, IAV4E, IAV2I	Flags for second-order explicit, fourth-order explicit, and second-order implicit artificial viscosity.
	IBCELM	Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the ξ and/or η directions; 0 if elimination is not necessary, 1 if it is.
*	ICHECK	Convergence checking interval.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	IT	Current time step number n.
	ITBEG	The time level n at the beginning of a run.
*	ITHIN	Flags for thin-layer option.
	KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
	NEQP	Dimensioning parameter specifying maximum number of coupled equations allowed.
	NMAXP	A dimensioning parameter equal to the maximum of N1P, N2P, and N3P.
	NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, N_1+1, N_2+1, and N_3+1 for spatially periodic boundary conditions in ξ, η, and ζ.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	N1P, N2P	Parameters specifying the dimension sizes in the ξ and η directions.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.

ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

DEL	Computational grid spacing in sweep direction.
IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
ISWEEP	Current ADI sweep number.
IV	Index in the "vectorized" direction, i_v.
I1, I2, I3	Grid indices i, j, and k, in the ξ, η, and ζ directions.
METX, METY, METZ, METT	Derivatives of sweep direction computational coordinate with respect to x, y, z, and t.
NPTS	Number of grid points in the sweep direction, N.
NV	Number of grid points in the "vectorized" direction, N_v.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level $n+1$.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
TL	Static temperature T at time level n.

Description

Subroutine EXEC manages the solution of the governing equations. It is called by the MAIN program during each marching step from time level n to $n+1$. The steps involved in EXEC are described below.

Preliminary Steps

1. If this is the first time step, temporarily set the thin-layer flags to zero.
2. Initialize the coefficient submatrices **A**, **B**, and **C**, and the source term subvector **S**, to zero.
3. If spatially periodic boundary conditions are being used in any direction, call PERIOD to add the appropriate extra line(s) of data.

First ADI sweep, ξ direction

4. Set various sweep-dependent parameters, as follows:

$$\begin{aligned} \texttt{isweep} &= 1 \\ \texttt{istep} &= 1 \\ \texttt{del} &= \Delta\xi \\ \texttt{nv} &= N_2 \text{ or } N_2+1 \end{aligned}$$

5. Begin loop in non-sweep (ζ) direction over interior points ($k = \text{I3} = 2$ to $\text{NPT3}-1$).
6. Set

$$\texttt{npts} = N_1 \text{ or } N_1 + 1$$

7. Set metrics in sweep (ξ) direction at all grid points as follows:

```
metx(i2,i1) = (ξx)i,j,k
mety(i2,i1) = (ξy)i,j,k
metz(i2,i1) = (ξz)i,j,k
mett(i2,i1) = (ξt)i,j,k
```

$$\texttt{metx(i2,i1)} = (\xi_x)_{i,j,k}$$
$$\texttt{mety(i2,i1)} = (\xi_y)_{i,j,k}$$
$$\texttt{metz(i2,i1)} = (\xi_z)_{i,j,k}$$
$$\texttt{mett(i2,i1)} = (\xi_t)_{i,j,k}$$

8. Begin loop in non-sweep (η) direction over interior points (j = I2 = 2 to NPT2 − 1).

9. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current η-ζ line at all ξ grid points.

10. Call the COEF routines to compute the coefficients and source terms for the governing equations along the current η-ζ line at all interior ξ grid points.

11. End of loop in non-sweep (η) direction.

12. For non-spatially periodic boundary conditions in the ξ direction, begin loop in non-sweep (η) direction over interior points (j = I2 = 2 to NPT2 − 1).

13. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current η-ζ line at all ξ grid points.

14. Call BCGEN to compute the coefficients and source terms for the boundary condition equations at the end points (i = I1 = 1 and N_1) of the current η-ζ line.

15. If three-point boundary conditions were used at either boundary, call BCELIM to eliminate the resulting off-diagonal coefficient submatrices.

16. End of loop in non-sweep (η) direction.

17. Every ICHECK time steps, call RESID to compute residuals at time level n without the artificial viscosity terms, and to update the convergence history file.

18. If artificial viscosity is being used, call AVISC1 or AVISC2 to add the appropriate terms to the coefficient submatrices and/or the source term subvectors at all interior grid points.

19. Every ICHECK time steps, if artificial viscosity is being used, call RESID to compute residuals at time level n with the artificial viscosity terms, and to update the convergence history file.

20. If spatially periodic boundary conditions are being used in the ξ direction, reset NPTS = N_1.

21. Call ADI to solve the system of difference equations.

22. Begin loop in non-sweep (η) direction over interior points (j = I2 = 2 to NPT2 − 1).

23. Call UPDATE to compute the primitive flow variables, $\mathbf{Q}^*$, from the newly computed conservation variables in delta form, $\Delta\hat{\mathbf{Q}}^*$, along the current η-ζ line at all ξ grid points.

24. End of loop in non-sweep (η) direction.

25. End of loop in non-sweep (ζ) direction.

Second ADI sweep, η direction

26. Set various sweep-dependent parameters, as follows:

$$\texttt{isweep} = 2$$
$$\texttt{istep} = \texttt{n1p}$$
$$\texttt{del} = \Delta\eta$$
$$\texttt{nv} = N_1 \text{ or } N_1 + 1$$

27. Begin loop in non-sweep (ζ) direction over interior points (k = I3 = 2 to NPT3 − 1).

28. Set

```
npts     = N2 or N2 + 1
```

29. Set metrics in sweep (η) direction at all grid points as follows:

$$
\begin{aligned}
\texttt{metx(i1,i2)} &= (\eta_x)_{i,j,k} \\
\texttt{mety(i1,i2)} &= (\eta_y)_{i,j,k} \\
\texttt{metz(i1,i2)} &= (\eta_z)_{i,j,k} \\
\texttt{mett(i1,i2)} &= (\eta_t)_{i,j,k}
\end{aligned}
$$

30. Begin loop in non-sweep (ξ) direction over interior points (i = I1 = 2 to NPT1 − 1).

31. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current ξ-ζ line at all η grid points.

32. Call the COEF routines to compute the coefficients and source terms for the governing equations along the current ξ-ζ line at all interior η grid points.

33. End of loop in non-sweep (ξ) direction.

34. For non-spatially periodic boundary conditions in the η direction, begin loop in non-sweep (ξ) direction over interior points (i = I1 = 2 to NPT1 − 1).

35. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current ξ-ζ line at all η grid points.

36. Call BCGEN to compute the coefficients and source terms for the boundary condition equations at the end points (j = I2 = 1 and N_2) of the current ξ-ζ line.

37. If three-point boundary conditions were used at either boundary, call BCELIM to eliminate the resulting off-diagonal coefficient submatrices.

38. End of loop in non-sweep (ξ) direction.

39. If implicit artificial viscosity is being used, call AVISC1 to add the appropriate terms to the coefficient submatrices at all interior grid points.

40. If spatially periodic boundary conditions are being used in the η direction, reset NPTS = N_2.

41. Call ADI to solve the system of difference equations.

42. Begin loop in non-sweep (ξ) direction over interior points (i = I1 = 2 to NPT1 − 1).

43. Call UPDATE to compute the primitive flow variables, $\mathbf{Q}^{**}$, from the newly computed conservation variables in delta form, $\Delta\hat{\mathbf{Q}}^{**}$, along the current ξ-ζ line at all η grid points.

44. End of loop in non-sweep (ξ) direction.

45. End of loop in non-sweep (ζ) direction.

Third ADI sweep, ζ direction

46. Set various sweep-dependent parameters, as follows:

```
isweep = 3
istep  = n1p*n2p
del    = Δζ
nv     = N1 or N1 + 1
```

47. Begin loop in non-sweep (η) direction over interior points (j = I2 = 2 to NPT2 − 1).

48. Set

```
npts     = N3 or N3 + 1
```

49. Set metrics in sweep (ζ) direction at all grid points as follows:

```
metx(i1,i3) = (ζx)i,j,k
mety(i1,i3) = (ζy)i,j,k
metz(i1,i3) = (ζz)i,j,k
mett(i1,i3) = (ζt)i,j,k
```

$$\texttt{metx(i1,i3)} = (\zeta_x)_{i,j,k}$$
$$\texttt{mety(i1,i3)} = (\zeta_y)_{i,j,k}$$
$$\texttt{metz(i1,i3)} = (\zeta_z)_{i,j,k}$$
$$\texttt{mett(i1,i3)} = (\zeta_t)_{i,j,k}$$

50. Begin loop in non-sweep (ξ) direction over interior points ($i = \mathrm{I1} = 2$ to NPT1 − 1).

51. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current ξ-η line at all ζ grid points.

52. Call the COEF routines to compute the coefficients and source terms for the governing equations along the current ξ-η line at all interior ζ grid points.

53. End of loop in non-sweep (ξ) direction.

54. For non-spatially periodic boundary conditions in the ζ direction, begin loop in non-sweep (ξ) direction over interior points ($i = \mathrm{I1} = 2$ to NPT1 − 1).

55. Call EQSTAT to get the derivatives of p and T with respect to ρ, ρu, etc., along the current ξ-η line at all ζ grid points.

56. Call BCGEN to compute the coefficients and source terms for the boundary condition equations at the end points ($k = \mathrm{I3} = 1$ and N_3) of the current ξ-η line.

57. If three-point boundary conditions were used at either boundary, call BCELIM to eliminate the resulting off-diagonal coefficient submatrices.

58. End of loop in non-sweep (ξ) direction.

59. If implicit artificial viscosity is being used, call AVISC1 to add the appropriate terms to the coefficient submatrices at all interior grid points.

60. If spatially periodic boundary conditions are being used in the ζ direction, reset NPTS = N_3.

61. Call ADI to solve the system of difference equations.

62. Begin loop in non-sweep (ξ) direction over interior points ($i = \mathrm{I1} = 2$ to NPT1 − 1).

63. Call UPDATE to compute the primitive flow variables, $\mathbf{Q}^{n+1}$, from the newly computed conservation variables in delta form, $\Delta\hat{\mathbf{Q}}^n$, along the current ξ-η line at all ζ grid points.

64. End of loop in non-sweep (ξ) direction.

65. End of loop in non-sweep (η) direction.

Finishing Steps

66. If this is the first time step, reset the thin-layer flags back to their input value.

67. Call BVUP to update the ξ and η boundary values, if necessary.

68. For all grid points, shift RHO and RHOL so that RHO = ρ^{n+1} and RHOL = ρ^n. Similarly, shift the Fortran variables for u, v, w, and E_T. Finally, set TL = T^n.

Subroutine EXECT		
Called by	Calls	Purpose
TURBCH	PERIOD SWDOWN SWUP UPDTKE	Manage solution of the k-ε equations.

Input

* CMUR	Constant C_{μ_r} in formula for C_μ.
* CTHREE	Constant C_3 in formula for C_μ.
E	Turbulent dissipation rate ε at time level n.
KBCPER	Flags for spatially periodic boundary conditions in the ξ and η directions; 0 for non-periodic, 1 for periodic.
KE	Turbulent kinetic energy k at time level n.
MUT	Turbulent viscosity μ_t at time level n.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
RHO	Static density ρ at time level n.
YPLUSD	Nondimensional distance y^+ from the nearest solid wall.

Output

EL	Turbulent dissipation rate ε at time level n.
KEL	Turbulent kinetic energy k at time level n.
MUT, MUTL	Turbulent viscosity μ_t at time levels $n + 1$ and n.

Description

Subroutine EXECT manages the solution of the k-ε equations. It is called by subroutine TURBCH, NTKE times per mean flow iteration. The steps involved in EXECT are described below.

1. If spatially periodic boundary conditions are being used in any direction, call PERIOD to add the appropriate extra line(s) of data.
2. Call SWUP to compute the coefficients and source terms for k-ε equations for the upward LU sweep, and to perform the sweep itself.
3. Call SWDOWN to compute the coefficients and source terms for k-ε equations for the downward LU sweep, and to perform the sweep itself.
4. For all grid points, set KEL $= k^n$ and EL $= \varepsilon^n$.
5. Call UPDTKE to compute the primitive flow variables k^{n+1} and ε^{n+1} from $\Delta\hat{\mathbf{W}}^n$, the newly computed conservation variables in delta form.
6. Compute the turbulent viscosity at each grid point, storing μ_t^{n+1} and μ_t^n in MUT and MUTL, respectively.

Subroutine FILTER		
Called by	Calls	Purpose
BLK4 BLK5	BLKOUT ISAMAX ISRCHEQ	Rearrange rows of the boundary condition coefficient submatrices and the source term subvector to eliminate any zeroes on the diagonal.

Input

A, B, C	Coefficient submatrices **A**, **B**, and **C** before rearrangement.
* IDEBUG	Debug flags.
* IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
ISWEEP	Current ADI sweep number.
IT	Current time step number n.
IV	Index in the "vectorized" direction, i_v.
NEQ	Number of coupled equations being solved, N_{eq}.
NMAXP	A dimensioning parameter equal to the maximum of N1P, N2P, and N3P.
* NOUT	Unit number for standard output.
NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
NPTS	Number of grid points in the sweep direction, N.
S	Source term subvector **S** before rearrangement.

Output

A, B, C	Coefficient submatrices **A**, **B**, and **C** after rearrangement.
S	Source term subvector **S** after rearrangement.

Description

Subroutine FILTER rearranges rows of the coefficient block submatrices and the source term subvector, at the two boundaries in the ADI sweep direction, in an attempt to eliminate any zero values on the diagonal of the submatrix **B**. These zero values may occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

For instance, if the specified initial conditions are zero velocity and constant flow properties everywhere in the flow field, the perfect gas equation of state yields:

$$E_T = \rho c_v T$$

$$p = (\gamma - 1)E_T$$

$$\frac{\partial p}{\partial \rho} = \frac{\partial p}{\partial(\rho u)} = \frac{\partial p}{\partial(\rho v)} = \frac{\partial p}{\partial(\rho w)} = 0$$

$$\frac{\partial p}{\partial E_T} = \gamma - 1$$

$$\frac{\partial T}{\partial \rho} = -\frac{E_T}{c_v \rho^2}$$

$$\frac{\partial T}{\partial(\rho u)} = \frac{\partial T}{\partial(\rho v)} = \frac{\partial T}{\partial(\rho w)} = 0$$

$$\frac{\partial T}{\partial E_T} = \frac{1}{c_v \rho}$$

If, in addition, the boundary conditions at a given boundary are, in order, specified pressure $p = f$, no-slip x-, y-, and z-velocity $u = v = w = 0$, and specified temperature $T = f$, then the linearization of the boundary conditions leads to the following **B** coefficient submatrix for that boundary:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & J(\gamma - 1) \\ 0 & J/\rho & 0 & 0 & 0 \\ 0 & 0 & J/\rho & 0 & 0 \\ 0 & 0 & 0 & J/\rho & 0 \\ -JE_T/c_v\rho^2 & 0 & 0 & 0 & J/c_v\rho \end{bmatrix}$$

The zero on the diagonal will lead to a divide-by-zero error in subroutine BLK5, even though this is not a singular matrix.

Subroutine FILTER tries to fix this problem. In this example, it finds a zero at element $\mathbf{B}_{11}$, searches column 1 for the largest element in absolute value (in this case $-JE_T/c_v\rho^2$), and adds that row to the row with the zero on the diagonal. Of course, the corresponding rows of **A**, **C**, and **S** must also be added together. The new **B** submatrix would be:

$$\mathbf{B} = \begin{bmatrix} -JE_T/c_v\rho^2 & 0 & 0 & 0 & J(\gamma - 1) + J/c_v\rho \\ 0 & J/\rho & 0 & 0 & 0 \\ 0 & 0 & J/\rho & 0 & 0 \\ 0 & 0 & 0 & J/\rho & 0 \\ -JE_T/c_v\rho^2 & 0 & 0 & 0 & J/c_v\rho \end{bmatrix}$$

Remarks

1. If a column with a zero on the diagonal has no other elements greater than 10^{-10}, then it is assumed that the matrix **B** is singular, which means the specified boundary conditions are not independent of one another. An error message is printed and the calculation is stopped.
2. It's probably sufficient to only call this subroutine for the first time step. After the first step, the chances of u, v, and w all being exactly zero at the same interior grid point are slim. Nevertheless, in the current version of *Proteus*, FILTER is called at every time step.
3. The Cray search routine ISAMAX is used in finding the largest element in any column corresponding to a zero on the matrix diagonal. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.
4. This subroutine generates the output for the IDEBUG(4) option.

Subroutine FTEMP		
Called by	Calls	Purpose
INITC MAIN		Compute auxiliary variables that are functions of temperature.

Input

	CCP1, CCP2, CCP3, CCP4	Constants in formula for specific heat.
	CK1, CK2	Constants in formula for laminar thermal conductivity coefficient.
	CMU1, CMU2	Constants in formula for laminar viscosity coefficient.
*	GAMR	Reference ratio of specific heats, γ_r.
	IGAM	Flag for constant or variable c_p, c_v, and γ; 0 if they are to be computed as functions of temperature, 1 if they are to be treated as constant.
*	ILAMV	Flag for computation of laminar viscosity and thermal conductivity.
*	NOUT	Unit number for standard output.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	RGAS	Gas constant R.
	T	Static temperature T.
*	TR, UR, MUR, KTR	Reference temperature T_r, velocity u_r, viscosity μ_r, and thermal conductivity k_r.

Output

CP, CV	Specific heats c_p and c_v.
MU, LA, KT	Laminar coefficient of viscosity μ_l, laminar second coefficient of viscosity λ_l, and laminar coefficient of thermal conductivity k_l.

Description

Subroutine FTEMP computes the auxiliary variables μ_l, λ_l, k_l, c_p, and c_v. For the laminar viscosities μ_l and λ_l, and the laminar thermal conductivity k_l, there are two options currently available.

If the input parameter ILAMV = 0 (the default), FTEMP sets the nondimensional values as:

$$\mu_l = 1$$
$$\lambda_l = -2/3$$
$$k_l = 1$$

Thus, with this option, the laminar viscosity and thermal conductivity are held constant at their reference values. These reference values may be specified by the user, or computed from the reference temperature. The laminar second coefficient of viscosity λ_l is set equal to $-2\mu_l/3$.

If ILAMV = 1, μ_l and k_l are computed as functions of temperature using Sutherland's formula (White, 1974). The formula for the laminar viscosity coefficient μ_l is

$$\mu_l = \frac{\overline{\mu}_l}{\mu_r} = \frac{\mu_r'}{\mu_r} \frac{T_r + C_{\mu 2}}{\overline{T} + C_{\mu 2}} \left(\frac{\overline{T}}{T_r} \right)^{3/2}$$

where the overbar indicates a dimensional value, and μ_r' is the laminar viscosity coefficient at $\overline{T} = T_r$, given by

$$\mu_r' = C_{\mu 1} \frac{T_r^{3/2}}{T_r + C_{\mu 2}}$$

Depending on the namelist input values of MUR and RER, μ_r' may or may not be equal to μ_r. These formulas are valid for air for temperatures from 300 to 3420 °R (167 to 1900 K). The value of the constants $C_{\mu 1}$ and $C_{\mu 2}$ depend on whether reference values are being specified by the user in English units (IUNITS = 0) or SI units (IUNITS = 1). The values are presented in Table 4-1. The laminar second coefficient of viscosity λ_l is set equal to $-2\mu_l/3$. The formula for the laminar thermal conductivity coefficient k_l is

$$k_l = \frac{\overline{k}_l}{k_r} = \frac{k_r'}{k_r} \frac{T_r + C_{k2}}{\overline{T} + C_{k2}} \left(\frac{\overline{T}}{T_r} \right)^{3/2}$$

where the overbar indicates a dimensional value, and k_r' is the laminar thermal conductivity coefficient at $\overline{T} = T_r$, given by

$$k_r' = C_{k1} \frac{T_r^{3/2}}{T_r + C_{k2}}$$

Depending on the namelist input values of KTR and PRLR, k_r' may or may not be equal to k_r. These formulas are valid for air for temperatures from 300 to 1800 °R (167 to 1000 K). The value of the constants C_{k1} and C_{k2} depend on whether reference values are being specified by the user in English units (IUNITS = 0) or SI units (IUNITS = 1). The values are presented in Table 4-1.

There are also two options available for the specific heat coefficients c_p and c_v. If the flag IGAM = 1, a value of the specific heat ratio γ has been specified by the user. In this case c_p and c_v are treated as constants, and computed from

$$c_v = \frac{R}{\gamma - 1}$$
$$c_p = c_v + R$$

If IGAM = 0, the user did not specify a value of γ. In this case, the specific heat coefficient c_p is computed as a function of temperature from the empirical formula of Hesse and Mumford (1964), and c_v is computed from that value assuming a thermally perfect gas. The ratio $\gamma = c_p/c_v$ will then vary with temperature. The equations for c_p and c_v are:

$$c_p = \overline{c}_p \frac{T_r}{u_r^2} = \frac{T_r}{u_r^2} \left(C_{c_p 1} - C_{c_p 2} \overline{T}^{-1/2} - C_{c_p 3} \overline{T} + C_{c_p 4} \overline{T}^2 \right)$$

$$c_v = c_p - R$$

This formula is valid for air for temperatures from 540 to 9000 °R (300 to 5000 K). The values of the constants $C_{c_p 1}$ through $C_{c_p 4}$ are presented in Table 4-1.

TABLE 4-1. - EMPIRICAL CONSTANTS FOR μ_l, k_l, AND c_p

CONSTANT	ENGLISH UNITS	SI UNITS
$C_{\mu 1}$	7.3035×10^{-7}	1.4582×10^{-6}
$C_{\mu 2}$	198.6	110.3
C_{k1}	7.4907×10^{-3}	1.8641×10^{-3}
C_{k2}	350.0	194.4
$C_{c_p 1}$	8.53×10^{3}	1.4264×10^{3}
$C_{c_p 2}$	3.12×10^{4}	3.8888×10^{3}
$C_{c_p 3}$	2.065×10^{6}	1.9184×10^{5}
$C_{c_p 4}$	7.83×10^{8}	4.0413×10^{7}

Remarks

1. The formulas used with the ILAMV = 1 option are for air. For other fluids, different formulas should be used to compute μ_l, λ_l, and k_l. These could easily be programmed as additional ILAMV options. Or, if the flow being computed is such that μ_l and k_l may be considered constant, simply set ILAMV = 0 and read in the appropriate values for μ_r and k_r. Note, however, that the ILAMV = 0 option still sets $\lambda_l = -2\mu_l/3$.
2. The formula used to compute c_p, when a value of γ is not specified by the user, is for air. For other gases, a different formula should be programmed. Or, if c_p and c_v may be considered constant, a value of γ should be read in.
3. An error message is generated and execution is stopped if an illegal value is specified for ILAMV.

Subroutine GATHER (N,VOUT,VIN,INDEX)		
Called by	Calls	Purpose
BLOUT		Create a vector containing specified elements of an input vector.

Input

N — Number of elements in the input vectors VIN and INDEX.

VIN — Input vector.

INDEX — Vector of indices specifying which elements of VIN are to be stored in VOUT.

Output

VOUT — Output vector containing elements of VIN specified by INDEX.

Description

Subroutine GATHER gathers a set of specified elements from an input vector and returns them in an output vector. The operation of GATHER is equivalent to the following Fortran code:

```
      do 10 i = 1,n
      vout(i) = vin(index(i))
10    continue
```

Remarks

1. GATHER is a Cray Linear Algebra routine (Cray Research, Inc., 1989b).

Subroutine GEOM		
Called by	Calls	Purpose
MAIN	METS PAK	Manage computation of grid and metric parameters.

Input

* IPACK	Flags for grid packing option.
* NGEOM	Flag for type of computational coordinates.
* NGRID	Unit number for input mesh file.
* NOUT	Unit number for standard output.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
N1P, N2P, N3P	Parameters specifying the dimension sizes in the ξ, η, and ζ directions.
* RMIN, RMAX	Minimum and maximum r-coordinates for cylindrical grid.
* THMIN, THMAX	Minimum and maximum θ-coordinates for cylindrical grid.
* XMIN, XMAX	Minimum and maximum x-coordinates for Cartesian or cylindrical grid.
* YMIN, YMAX	Minimum and maximum y-coordinates for Cartesian grid.
* ZMIN, ZMAX	Minimum and maximum z-coordinates for Cartesian grid.

Output

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
X, Y, Z	Cartesian coordinates x, y, and z.

Description

Subroutine GEOM manages the computation of the grid and metric parameters. There are currently three coordinate system options built into *Proteus*, as follows:

NGEOM	Computational Coordinates
1	Cartesian (x-y-z)
2	Cylindrical (r-θ-x)
10	Read from separate file.

Subroutine GEOM first computes the grid spacing in computational space in the ξ, η, and ζ directions as $\Delta\xi = 1/(N_1 - 1)$, $\Delta\eta = 1/(N_2 - 1)$, and $\Delta\zeta = 1/(N_3 - 1)$. Note that grid points in computational space are always evenly distributed along the (ξ-η-ζ) coordinate lines.

Cartesian (x-y-z) Coordinates (NGEOM = 1)

For the Cartesian coordinate option, an evenly spaced set of physical Cartesian (x-y-z) coordinates are related to the computational (ξ-η-ζ) coordinates by

$$x = x_{min} + (x_{max} - x_{min})\xi$$
$$y = y_{min} + (y_{max} - y_{min})\eta$$
$$z = z_{min} + (z_{max} - z_{min})\zeta$$

If grid packing is used, subroutine PAK is called to redistribute these points according to the packing parameters specified by the user, and to interpolate to get the new physical Cartesian (x-y-z) coordinates in the computational mesh. Subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

Cylindrical (r-θ-x) Coordinates (NGEOM = 2)

For the cylindrical coordinate option, an evenly spaced set of physical cylindrical (r-θ-x) coordinates are related to the computational (ξ-η-ζ) coordinates by

$$\theta = \theta_{min} + (\theta_{max} - \theta_{min})\xi$$
$$r = r_{min} + (r_{max} - r_{min})\eta$$
$$x = x_{min} + (x_{max} - x_{min})\zeta$$

The Cartesian (x-y-z) coordinates are simply given by

$$x = x$$
$$y = r \sin\theta$$
$$z = r \cos\theta$$

As in the NGEOM = 1 option, if grid packing is used, subroutine PAK is called to redistribute these points according to the packing parameters specified by the user, and to interpolate to get the new physical Cartesian (x-y-z) coordinates in the computational mesh. Subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

Coordinates Read From Separate File (NGEOM = 10)

The third option for specifying the computational coordinate system is to read it from a separate file, as described in Section 3.2 of Volume 2. The computational (ξ-η-ζ) coordinate system is determined by a set of $N_{G1} \times N_{G2} \times N_{G3}$ points whose physical Cartesian (x-y-z) coordinates are specified. Here N_{G1}, N_{G2}, and N_{G3} are the number of points in the ξ, η, and ζ directions used to specify the computational coordinate system. Note that they do not have to be equal to N_1, N_2, and N_3, the number of points in the computational mesh used for the finite-difference method.[24] Note also that the points do not have to be equally distributed in physical space along the ξ, η, and ζ coordinate lines.

If grid packing is being used, subroutine PAK is called to distribute $N_1 \times N_2 \times N_3$ computational mesh points in physical space according to the packing parameters SQ specified by the user, and to interpolate among the $N_{G1} \times N_{G2} \times N_{G3}$ points in the input computational coordinate system to get the new physical Cartesian coordinates of the points in the computational mesh.

If grid packing is not being used, but N_{G1}, N_{G2}, and N_{G3} are not equal to N_1, N_2, and N_3 respectively, then subroutine PAK is still called. In this case, however, PAK distributes the $N_1 \times N_2 \times N_3$ computational mesh points evenly in physical space and then interpolates among the $N_{G1} \times N_{G2} \times N_{G3}$ points in the input computational coordinate system to get the new physical Cartesian coordinates of the points in the computational mesh.

In either case, subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

[24] The distinction between the computational coordinate system and the computational mesh is described in Section 2.2 of Volume 2.

Remarks

1. An error message is generated and execution is stopped if an illegal coordinate system option is specified.
2. With the NGEOM = 10 option, an error message is generated and execution is stopped if N_{G1}, N_{G2}, and/or N_{G3} are greater than the dimensioning parameters N1P, N2P, and/or N3P.

Subroutine INIT		
Called by	Calls	Purpose
INITC		Get user-defined initial flow field.

Input

* ICVARS	Flag specifying which variables are being supplied as initial conditions by subroutine INIT.
NIN	Unit number for namelist input.
* NOUT	Unit number for standard output.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

Output

P, T, U, V, W	Initial flow field values of static pressure p, static temperature T, and velocities u, v, and w.

Description

Subroutine INIT supplies the user-defined initial flow field. In general, this subroutine will be tailored to the problem being solved, and supplied by the user. Details on the variables to be supplied by INIT are presented in Section 5.1 of Volume 2.

A default version of INIT is supplied with *Proteus* that specifies uniform flow with constant properties everywhere in the flow field. The above list of input and output Fortran variables are for the default version of INIT. The default version assumes ICVARS = 2 (the default value), and reads values of p_0, u_0, v_0, w_0, and T_0 from namelist IC. The defaults for these parameters are 1.0, 0.0, 0.0, 0.0, and 1.0, respectively, resulting in an initial flow field with $\bar{p} = p_r$, $u = v = w = 0$, and $\bar{T} = T_r$.

Remarks

1. If a value for ICVARS other than 2 is set in the input, a warning message is generated and ICVARS is reset to 2.
2. Subroutine INIT is a convenient place to specify point-by-point boundary condition types and values. It's often easier to do this using Fortran coding rather than entering each value into the namelist input file.

Subroutine INITC		
Called by	Calls	Purpose
MAIN	EQSTAT FTEMP INIT KEINIT REST TURBBL YPLUSN	Set up consistent initial conditions based on data from INIT.

Input

* CMUR	Constant C_{μ_r} in formula for C_μ.
CP	Specific heat c_p.
* CTHREE	Constant C_3 in formula for C_μ.
EP1	Minimum allowable numerical value.
* GAMR	Reference ratio of specific heats, γ_r.
GC	Proportionality factor g_c in Newton's second law.
* HSTAG	Stagnation enthalpy h_T used with constant stagnation enthalpy option.
* ICVARS	Flag specifying which variables are being supplied as initial conditions by subroutine INIT.
* IHSTAG	Flag for constant stagnation enthalpy option.
* IREST	Flag for reading/writing restart file.
ITBEG	The time level n at the beginning of a run.
* ITURB	Flag for turbulent flow option.
* KBC1, KBC2, KBC3	Boundary types for the ξ, η, and ζ directions.
LWSET	Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
MU, LA, KT	Laminar coefficient of viscosity μ_l, laminar second coefficient of viscosity λ_l, and laminar coefficient of thermal conductivity k_l.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
PR	Reference pressure p_r.
PRR	Reference Prandtl number Pr_r.
* PRT	Turbulent Prandtl number Pr_t, or, if PRT ≤ 0, a flag indicating the use of a variable turbulent Prandtl number.
RGAS	Gas constant R.
* RHOR, UR	Reference density ρ_r and velocity u_r.
INITIAL FLOW FIELD	From the user-suppled or default version of subroutine INIT. The combination of variables supplied by INIT is specified by ICVARS. See Section 5.0 of Volume 2 for details.

Output

LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries, if not set in input.
MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k.
MUT, MUTL	Turbulent viscosity μ_t at time levels n and $n-1$.
RHO, U, V, W, ET	Initial flow field values of static density ρ, velocities u, v, and w, and total energy E_T at time level n.
RHOL, UL, VL, WL, ETL	Initial flow field values of static density ρ, velocities u, v, and w, and total energy E_T at time level $n-1$.
TL	Static temperature T at time level $n-1$.

Description

Subroutine INITC sets up consistent initial flow field conditions based on the data supplied by subroutine INIT. For restart cases, subroutine REST is called to read the computational mesh and the initial flow field. Otherwise, the data supplied by INIT are used to obtain the density ρ, the velocities u, v, and w, and the temperature T.[25] It then calls FTEMP to compute the laminar viscosity coefficients μ_l and λ_l, the laminar thermal conductivity coefficient k_l, and the specific heat coefficients c_p and c_v. EQSTAT is called next to compute the pressure p and to recompute the temperature T.[26] For turbulent flow, the appropriate subroutines are called to compute the effective viscosity and thermal conductivity coefficients using the turbulence model specified by the user. And finally, for non-restart cases, the values of the dependent variables at time level $n-1$ are set equal to the values at level 1.

The flag ICVARS is used to specify which combination of variables are being supplied by INIT. The calculation of ρ, u, v, w, and T is described below for the different values of ICVARS. In all of the equations below, the specific heats are defined by

$$c_v = \frac{R}{\gamma_r - 1}$$

$$c_p = R + c_v$$

where γ_r is either specified by the user or computed from the reference temperature T_r.

ICVARS = 1

With this option, the density ρ, the momentum components ρu, ρv, and ρw, and if IHSTAG = 0 the total energy E_T, are supplied by INIT. Thus, the velocity components are simply

$$u = \frac{\rho u}{\rho}$$

$$v = \frac{\rho v}{\rho}$$

$$w = \frac{\rho w}{\rho}$$

If the energy equation is being solved (IHSTAG = 0), the temperature is computed from

[25] The calculation of T at this point may be approximate. See Remark 1.

[26] See Remark 1.

$$T = \frac{1}{c_v}\left[\frac{E_T}{\rho} - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

If the energy equation is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

$$T = \frac{1}{c_p}\left[h_T - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

ICVARS = 2

With this option, the pressure p and the velocities u, v, and w are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the temperature T is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

$$T = \frac{1}{c_p}\left[h_T - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

The density is then given by

$$\rho = \frac{p}{RT}$$

and the total energy is

$$E_T = \rho\left[c_v T + \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

ICVARS = 3

With this option, the density ρ and the velocities u, v, and w are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the temperature T is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

$$T = \frac{1}{c_p}\left[h_T - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

The total energy is then

$$E_T = \rho\left[c_v T + \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

ICVARS = 4

With this option, the pressure p and the velocities u, v, and w are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the density ρ is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), this option is the same as the ICVARS = 2 option. If the energy equation is being solved, then, the temperature is

$$T = \frac{p}{\rho R}$$

The total energy is then

$$E_T = \rho\left[c_v T + \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

ICVARS = 5

With this option, the static pressure coefficient c_p and the velocities u, v, and w are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the temperature T is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

$$T = \frac{1}{c_p}\left[h_T - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

The pressure coefficient is defined by

$$c_p = \frac{(\bar{p} - p_r)g_c}{\rho_r u_r^2/2}$$

The nondimensionalized pressure $p = \bar{p}g_c/\rho_r u_r^2$ is thus

$$p = \frac{c_p}{2} + \frac{p_r g_c}{\rho_r u_r^2}$$

or, since $p_r = \rho_r \bar{R} T_r/g_c$ and the nondimensionalized gas constant $R = \bar{R}T_r/u_r^2$,

$$p = \frac{c_p}{2} + R$$

The density is then

$$\rho = \frac{p}{RT}$$

and the total energy is

$$E_T = \rho\left[c_v T + \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

ICVARS = 6

With this option, the pressure p, Mach number M, and flow angles α_v and α_w are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the temperature T is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

$$T = T_T\left(1 + \frac{\gamma_r - 1}{2}M^2\right)^{-1}$$

where $T_T = h_T/c_p$. The density is

$$\rho = \frac{p}{RT}$$

The flow angles are defined by $\alpha_v = \tan^{-1}(v/u)$ and $\alpha_w = \tan^{-1}(w/u)$. The Mach number is defined by

$$M = \left(\frac{u^2 + v^2 + w^2}{\gamma_r RT}\right)^{1/2}$$

Solving for u,

$$u = M\left[\frac{\gamma_r R T}{1 + (v/u)^2 + (w/u)^2}\right]^{1/2}$$

where $(v/u)^2 = \tan^2\alpha_v$ and $(w/u)^2 = \tan^2\alpha_w$. The remaining velocities are simply

$$v = u \tan \alpha_v$$

$$w = u \tan \alpha_w$$

The total energy is

$$E_T = \rho\left[c_v T + \frac{1}{2}(u^2 + v^2 + w^2)\right]$$

Remarks

1. If T is not supplied by INIT, it must be computed from the equation of state. The equation of state contains a specific heat coefficient (either c_p or c_v, depending on whether the stagnation enthalpy is assumed constant or not.) The first time T is computed in INITC, a constant value of specific heat is used, consistent with the reference temperature T_r. If the user specified constant specific heat (i.e., a value for γ_r was read in), this is not a problem. However, if the temperature-dependent specific heat option is being used (i.e., a value for γ_r was not read in), the equation of state and the empirical equation for specific heat are coupled. For this reason T is recomputed in EQSTAT after the specific heats are computed in FTEMP. Ideally, this coupling would be handled by iteration between FTEMP and EQSTAT. This is not currently done in *Proteus*, however.
2. For options in which the pressure p is specified (ICVARS = 2, 4, and 6), the value supplied by INIT is redefined as follows:

$$p = p\frac{p_r g_c}{\rho_r u_r^2}$$

   This is necessary because input and output values of p are nondimensionalized by the reference pressure $p_r = \rho_r \bar{R} T_r$, while internal to the code itself p is nondimensionalized by the normalizing pressure $p_n = \rho_r u_r^2$. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.
3. With the ICVARS = 6 option, the initial velocity u will be limited to non-negative values.
4. If non-positive pressures or temperatures were computed in EQSTAT, the Fortran variable INEG will be positive. An error message will be printed, including a table showing the location of the non-positive values. The calculation will stop in INITC.
5. An error message is generated and execution is stopped if an illegal value is specified for ICVARS.
6. An error message is generated and execution is stopped if the value of ITURB does not correspond to an existing turbulence model.

Subroutine INPUT		
Called by	Calls	Purpose
MAIN	ISAMAX	Read and print input, perform various initializations.

Input

NIN	Unit number for namelist input.
NTP	Dimensioning parameter specifying the maximum number of entries in the table of time-dependent boundary condition values.
NTSEQP	Dimensioning parameter specifying the maximum number of time step sequences for the time step sequencing option.
N1P, N2P, N3P	Parameters specifying the dimension sizes in the ξ, η, and ζ directions.

Output

CKMIN	Constant $(C_{Kleb})_{min}$ in the Klebanoff intermittency factor.
GAMR	Reference ratio of specific heats, γ_r.
HSTAG, HSTAGR	Dimensionless and dimensional stagnation enthalpy h_T for the constant stagnation enthalpy option.
IGAM	Flag for constant or variable c_p, c_v, and γ; 0 if they are to be computed as functions of temperature, 1 if they are to be treated as constant.
* IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
ITDBC	Flag for time-dependent boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used.
LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries.
LWSET	Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
MACHR	Reference Mach number M_r.
MUR, KTR	Reference viscosity coefficient μ_r and thermal conductivity coefficient k_r.
NEQ	Number of coupled equations being solved, N_{eq}.
NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
PR	Reference pressure p_r.
PRLR	Reference laminar Prandtl number Pr_{l_r}.
RER, PRR	Reference Reynolds number Re_r and Prandtl number Pr_r.
RGAS	Gas constant R.
UR	Reference velocity u_r.

Description

Subroutine INPUT performs various input and initialization functions. It first reads the title and namelist input from the standard input file. Namelist RSTRT is read first, followed by namelist IO. If IUNITS = 1, indicating reference conditions will be specified in SI units, various default values and constants are redefined to be consistent with SI units. The remaining namelists are then read.

Next, the flags controlling the time step cycling and the convergence testing method are redefined, if necessary, to be consistent with each other. The number of equations being solved is then determined based on the values of IHSTAG. A flag is set if time-dependent boundary conditions are being used. The LWSET flags, which specify how wall locations are to be determined for the turbulence model, are defined based on the default and input values of the LWALL and IWALL parameters. If the user did not specify a value for $(C_{kleb})_{min}$, it is set to the default value, which depends on the turbulence model being used.

Next, if frequency of printout in the ξ, η, and ζ directions is being set by the input arrays IPRT1, IPRT2, and IPRT3, the corresponding grid indices are stored in arrays IPRT1A, IPRT2A, and IPRT3A. The total number of printout locations in each direction is also determined.

A header is then written to the standard output file, followed by the input namelists. Note that, for variables not specified by the user in the input namelists, the values in this printout will be the default values.

Various checks are made for inconsistent or invalid input, and appropriate error or warning messages are written. These are described in Section 7.0 of Volume 2.

Next, any reference or normalizing conditions not already defined are calculated. The reference and normalizing conditions are then written to the standard output file, with the appropriate units. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.

Remarks

1. The Cray search routine ISAMAX is used in the input consistency check to determine whether any implicit artificial viscosity coefficients are non-zero.

Function ISAMAX (N,V,INC)		
Called by	Calls	Purpose
BLOUT CONV FILTER INPUT RESID TIMSTP		Find the first index corresponding to the largest absolute value of the elements of a Fortran vector.

Input

N — Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).

V — Vector to be searched.

INC — Skip distance between elements of V. For contiguous elements, INC = 1.

Output

ISAMAX — First index corresponding to the largest absolute value of the elements of V that were searched.

Description

Function ISAMAX finds the first index corresponding to the largest absolute value of the elements of a vector. For a one-dimensional vector, the use of ISAMAX is straightforward. For example,

```
imax = isamax(np,v,1)
```

sets IMAX equal to the index I corresponding to the maximum value of V(I) for I = 1 to NP.

A starting location can be specified, as in

```
imax = 4 + isamax(np-4,v(5),1)
```

which sets IMAX equal to the index I corresponding to the maximum value of V(I) for I = 5 to NP.

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2 by NDIM3, then

```
imax = isamax(ndim1*ndim2*ndim3,a,1)
```

sets IMAX equal to the one-dimensional index corresponding to the maximum value of A(I,J,K) for all I, J, and K. The maximum value of A can then be referenced as A(IMAX,1,1).

One dimension at a time can also be searched. For example,

```
imax = isamax(ndim1,a(1,5,1),1)
```

sets IMAX equal to the index I corresponding to the maximum value of A(I,5,1) for I varying from 1 to NDIM1. Similarly, by specifying a skip increment,

```
jmax = isamax(ndim2,a(5,j,1),ndim1)
```

sets JMAX equal to the index J corresponding to the maximum value of A(5,J,1) for J varying from 1 to NDIM2.

Remarks

1. ISAMAX is a Cray search routine (Cray Research, Inc., 1989b).

Function ISAMIN (N,V,INC)		
Called by	Calls	Purpose
BLOUT OUTPUT TIMSTP		Find the first index corresponding to the smallest absolute value of the elements of a Fortran vector.

Input

N	Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).
V	Vector to be searched.
INC	Skip distance between elements of V. For contiguous elements, INC = 1.

Output

ISAMIN	First index corresponding to the smallest absolute value of the elements of V that were searched.

Description

Function ISAMIN finds the first index corresponding to the smallest absolute value of the elements of a vector. It is used in exactly the same way as ISAMAX.

Remarks

1. ISAMIN is a Cray search routine (Cray Research, Inc., 1989b).

Function ISRCHEQ (N,V,INC,VALUE)		
Called by	Calls	Purpose
BCGEN FILTER		Find the first index in a vector whose element is equal to a specified value.

Input

N — Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).

V — Vector to be searched.

INC — Skip distance between elements of V. For contiguous elements, INC = 1.

VALUE — Value to be searched for in the vector V.

Output

ISRCHEQ — First index, of the elements of V that were searched, whose element is equal to the value V. If the value V is not found, the returned value will be N + 1.

Description

Function ISRCHEQ finds the first index in a vector whose element is equal to a specified value. For a one-dimensional vector, the use of ISRCHEQ is straightforward. For example,

```
ival = isrcheq(np,v,1,val)
```

searches V(I), for I = 1 to NP, for the value VAL, and sets IVAL equal to the first index I for which V(I) = VAL. If the value VAL is not found, IVAL will be equal to NP + 1.

A starting location can be specified, as in

```
ival = 4 + isrcheq(np-4,v(5),1,val)
```

which searches V(I), for I = 5 to NP, for the value VAL.

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2 by NDIM3, then

```
ival = isrcheq(ndim1*ndim2*ndim3,a,1,val)
```

searches A(I,J,K), for all I, J, and K, for the value VAL, and sets IVAL equal to the corresponding one-dimensional index. The desired indices in A can then be recovered from

```
i = mod(ival-1,ndim1) + 1
j = mod(ival-1,ndim1*ndim2)/ndim1 + 1
k = (ival-1)/(ndim1*ndim2) + 1
```

One dimension at a time can also be searched. For example,

```
ival = isrcheq(ndim1,a(1,5,1),1,val)
```

searches A(I,5,1), for I = 1 to NDIM1, for the value VAL. Similarly, by specifying a skip increment,

```
jval = isrcheq(ndim2,a(5,j,1),ndim1,val)
```

searches A(5,J,1), for J = 1 to NDIM2, for the value VAL.

Remarks

1. ISRCHEQ is a Cray search routine (Cray Research, Inc., 1989b).

Function ISRCHFGT (N,V,INC,VALUE)		
Called by	Calls	Purpose
BLIN BLOUT		Find the first index in an array whose element is greater than a specified value.

Input

N — Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).

V — Vector to be searched.

INC — Skip distance between elements of V. For contiguous elements, INC = 1.

VALUE — Value to be searched for in the vector V.

Output

ISRCHFGT — First index, of the elements of V that were searched, whose element is greater than the value V. If the value V is not found, the returned value will be N + 1.

Description

Function ISRCHFGT finds the first index in a vector whose element is greater than a specified value. It is used in exactly the same way as ISRCHEQ.

Remarks

1. ISRCHFGT is a Cray search routine (Cray Research, Inc., 1989b).

Function ISRCHFLT (N,V,INC,VALUE)		
Called by	Calls	Purpose
BLOUT		Find the first index in an array whose element is less than a specified value.

Input

N	Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).
V	Vector to be searched.
INC	Skip distance between elements of V. For contiguous elements, INC = 1.
VALUE	Value to be searched for in the vector V.

Output

ISRCHFLT	First index, of the elements of V that were searched, whose element is less than the value V. If the value V is not found, the returned value will be N + 1.

Description

Function ISRCHFLT finds the first index in a vector whose element is less than a specified value. It is used in exactly the same way as ISRCHEQ.

Remarks

1. ISRCHFLT is a Cray search routine (Cray Research, Inc., 1989b).

Subroutine KEINIT		
Called by	Calls	Purpose
INITC	PRODCT TURBBL YPLUSN	Get user-defined initial conditions for k and ε.

Input

*	CMUR	Constant C_{μ_r} in formula for C_μ.
*	CTHREE	Constant C_3 in formula for C_μ.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
*	RER	Reference Reynolds number Re_r.
	RHO	Static density ρ at time level n.

Output

E, EL	Turbulent dissipation rate ε at time levels n and $n-1$.
KE, KEL	Turbulent kinetic energy k at time levels n and $n-1$.
MUTL	Turbulent viscosity μ_t at time level $n-1$.

Description

Subroutine KEINIT supplies the user-defined initial values of the turbulent kinetic energy k and the turbulent dissipation rate ε. In general, this subroutine will be tailored to the problem being solved, and supplied by user. Details on the variables to be supplied by KEINIT are presented in Section 5.1 of Volume 2.

A default version of KEINIT is supplied with *Proteus* that computes the initial values of k and ε using the assumption of local equilibrium (dissipation equals production.) The above list of input and output Fortran variables are for the default version of KEINIT.

The steps involved in the default version of KEINIT are described below.

1. Initialize k and ε to zero.
2. Call TURBBL to compute turbulent viscosity values and to locate solid walls in the computational domain.
3. Call YPLUSN to compute y^+ and the minimum distance to the nearest solid wall.
4. Call PRODCT to compute the production rate of turbulent kinetic energy.
5. Compute k and ε using

$$C_\mu = C_{\mu_r}\left(1 - e^{C_3 y^+}\right)$$

$$\varepsilon = \left| \frac{P_k}{Re_r \rho} \right|$$

$$k = \sqrt{\frac{\mu_t \varepsilon}{C_\mu \rho}}$$

6. Set the values of k, ε, and μ_t at time level $n - 1$ equal to their values at time level n.

Remarks

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the distance to the nearest wall. The array is filled in subroutine YPLUSN.
2. The Fortran array VORT, from the common block TURB1, is used to store the values of the production rate of turbulent kinetic energy. The array is filled in subroutine PRODCT.

Program MAIN		
Called by	Calls	Purpose
	BCSET CONV EQSTAT EXEC FTEMP GEOM INITC INPUT OUTPUT OUTW PLOT PRTHST REST TBC TIMSTP TREMAIN TURBBL TURBCH	Manage overall solution.

Input

None.

Output

IT	Current time step number n.
ITEND	Final time step number.
ITSEQ	Current time step sequence number.
TAU	Current time value τ.

Description

The MAIN program is used to manage the overall solution. The steps involved are described below.

Preliminary Steps

1. Call INPUT to read and print the input, and perform various initialization procedures.
2. Unless this is a restart case, call GEOM to get the computational coordinates and metric data.
3. Call INITC to get the initial flow field.
4. Call BCSET to set various boundary condition parameters and flags, and to print the input boundary condition types and values.
5. Initialize the plot file,[27] and, if requested by the user, write the initial or restart flow field into the plot file.
6. If requested by the user, print the initial or restart flow field.

---

[27] The initialization procedure depends on the type of plot file being written. See the description of subroutine PLOT.

7. Compute NTSUM, the maximum total number of marching steps to be taken, and ITEND, the corresponding final index on the time marching loop. Set the initial values of ITSEQ, the time step sequence number, and ITSWCH, the time index for switching to the next sequence, both to zero.

Time marching loop

8. Begin the time marching loop. The loop index IT corresponds to the known time level n. Each iteration of the loop thus corresponds to a step from time level n to $n + 1$.

9. If at the end of a time step sequence, update ITSEQ, the time step sequence number, and ITSWCH, the time index for switching to the next sequence.

10. For the first time step, and every IDTMOD'th step thereafter, call TIMSTP to compute the new time step $\Delta\tau$. For every time step update the time value τ.

11. If time-dependent boundary conditions are being used, call TBC to set the boundary condition values.

12. Call EXEC to solve the equations.

13. Call EQSTAT to compute the pressure p and temperature T from the equation of state. If either is non-positive, indicating a non-physical solution, skip forward to step 17.

14. Call FTEMP to compute the laminar viscosities μ_l and λ_l, the laminar thermal conductivity k_l, and the specific heats c_p and c_v.

15. For turbulent flow, call the appropriate subroutines to compute the effective viscosity and thermal conductivity coefficients using the turbulence model specified by the user.

16. Every ICHECK time levels, call CONV to check for convergence.

17. Call TREMAIN to find out how much CPU time remains.

18. If requested by the user, or if the calculation is converged, or if non-positive pressures or temperatures were computed, or if the job is near the CPU time limit, print the flow field at time level $n + 1$.

19. If requested by the user, or if the calculation is converged, or if non-positive pressures or temperatures were computed, or if the job is near the CPU time limit, write the flow field at time level $n + 1$ into the plot file.

20. If non-positive pressures or temperatures were computed, write an error message showing the location of the non-positive values and skip forward to step 25, ending the calculation.

21. If the calculation is converged, print a message and skip forward to step 24, ending the calculation.

22. If the job is near the CPU time limit, print a message and skip forward to step 24, ending the calculation.

23. End of time marching loop. Print a message indicating the calculation did not converge.

Final Steps

24. If requested by the user, call REST to write the restart file.

25. If first-order time differencing and steady boundary conditions were used, call PRTHST to print the convergence history.

Remarks

1. The starting index for the time marching loop is ITBEG. For a non-restart case ITBEG = 1, and thus the initial starting flow field is at time level 1. For a restart case ITBEG = n, where n is the time level stored in the restart file, and thus the starting flow field is the previously computed flow field at time level n.

2. The ending index for the time marching loop is ITEND = ITBEG + NTSUM − 1, where NTSUM is the total number of time steps to be taken. For a non-restart case, then, the time marches from level

1 to level 1 + NTSUM. For a restart case, the time marches from level ITBEG to level ITBEG + NTSUM.

3. The logic involving NTSUM, ITSEQ, and ITSWCH is used to implement the time step sequencing option. This allows one CFL number or time increment to be used for a specified number of steps, followed by another CFL number or time increment for another specified number of steps, etc.[28] If this option is not used, NTSUM is simply equal to NTIME(1) and ITSEQ is always 1.

4. An error message is generated and execution is stopped if the value of ITURB does not correspond to an existing turbulence model.

5. Although the calculation will stop if p or $T \leq 0$, as noted above in step 19, the standard output and plot file will include the time level with the non-positive values, if that is consistent with the IPRT and IPLT input parameters in namelist IO. The restart file will not be written.

[28] See Section 3.1.9 of Volume 2 for details on how to invoke the time step sequencing option.

Subroutine METS		
Called by	Calls	Purpose
GEOM REST	OUTPUT	Compute metrics of nonorthogonal grid transformation.

Input

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
* IDEBUG	Debug flags.
* IVOUT	Flags specifying variables to be printed.
* NOUT	Unit number for standard output.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
X, Y, Z	Cartesian coordinates x, y, and z.

Output

ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
IVOUT	Flags specifying variables to be printed (temporarily redefined for debug output of metrics.)
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Description

Subroutine METS computes the metric coefficients and the Jacobian for the generalized nonorthogonal coordinate transformation. The metric coefficients are defined in terms of the known (x-y-z) coordinates of the computational mesh as:

$$\xi_x = J[(y_\eta z)_\zeta - (y_\zeta z)_\eta]$$

$$\xi_y = J[(x_\zeta z)_\eta - (x_\eta z)_\zeta]$$

$$\xi_z = J[(x_\eta y)_\zeta - (x_\zeta y)_\eta]$$

$$\eta_x = J[(y_\zeta z)_\xi - (y_\xi z)_\zeta]$$

$$\eta_y = J[(x_\xi z)_\zeta - (x_\zeta z)_\xi]$$

$$\eta_z = J[(x_\zeta y)_\xi - (x_\xi y)_\zeta]$$

$$\zeta_x = J[(y_\xi z)_\eta - (y_\eta z)_\xi]$$

$$\zeta_y = J[(x_\eta z)_\xi - (x_\xi z)_\eta]$$

$$\zeta_z = J[(x_\xi y)_\eta - (x_\eta y)_\xi]$$

$$\xi_t = -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z$$

$$\eta_t = -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z$$

$$\zeta_t = -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z$$

where J is the Jacobian of the transformation, given by

$$J = \frac{1}{J^{-1}} = [x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) + x_\eta(y_\zeta z_\xi - y_\xi z_\zeta) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)]^{-1}$$

The derivatives of x, y, and z with respect to the computational coordinates are computed numerically using the same difference formulas as used for the governing equations. At interior points the centered difference formula presented in Section 5.0 of Volume 1 is used. At boundaries three-point one-sided differencing is used. For ξ-derivatives at the $\xi = 0$ and $\xi = 1$ boundaries,

$$\frac{\partial f}{\partial \xi} \simeq \pm \frac{-3f_w + 4f_{w\pm 1} - f_{w\pm 2}}{2\Delta\xi}$$

where w represents the ξ-index at the boundary (i.e., either 1 or N_1). Where a $\pm$ sign appears, the $+$ sign is used at the $\xi = 0$ boundary, and the $-$ sign is used at the $\xi = 1$ boundary. An analogous formula is used for η-derivatives at the $\eta = 0$ and $\eta = 1$ boundaries, and for ζ-derivatives at the $\zeta = 0$ and $\zeta = 1$ boundaries.

Remarks

1. Several local three-dimensional Fortran arrays (XXI, XETA, etc.), are used in METS to store the derivatives x_ξ, x_η, etc. These arrays are equivalenced to flow variables from common block FLOW1, which, at the point METS is called, have not yet been assigned values. These flow variables are set equal to zero at the end of METS.
2. Since the current version of *Proteus* is limited to meshes that do not vary with time, the derivatives x_τ, y_τ, and z_τ are set equal to zero.
3. This subroutine generates the output for the IDEBUG(7) option.
4. An error message is generated and execution is stopped if the grid transformation Jacobian J changes sign or equals zero. This indicates that the computational mesh contains crossed or coincident grid lines. The error message is followed by a printout of the Cartesian coordinates, the Jacobian, and the metric coefficients.

Subroutine OUTPUT (LEVEL)		
Called by	Calls	Purpose
MAIN METS	ISAMIN PRTOUT VORTEX	Manage printing of output.

Input

	CP, CV	Specific heats c_p and c_v.
	DTAU	Time step $\Delta\tau$.
	DUMMY	A three-dimensional scratch array.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	E, KE	Turbulent dissipation rate ε and kinetic energy k.
	ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
*	GAMR	Reference ratio of specific heats, γ_r.
	GC	Proportionality factor g_c in Newton's second law.
*	IVOUT	Flags specifying variables to be printed.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	LEVEL	Time level to be printed.
	LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries.
*	MACHR	Reference Mach number M_r.
	MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k.
	MUT	Turbulent viscosity coefficient μ_t.
*	NOUT	Unit number for standard output.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	P, T	Static pressure p and temperature T.
	PR	Reference pressure p_r.
	PRR	Reference Prandtl number Pr_r.
*	PRT	Turbulent Prandtl number Pr_t.
	RGAS	Gas constant R.
	RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T.
*	RHOR, TR, UR	Reference density ρ_r, temperature T_r, and velocity u_r.
	TAU	Time value τ.
	X, Y, Z	Cartesian coordinates x, y, and z.
	XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
	ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

ATITLE	A 20-character title for variable being printed.
DUMMY	A three-dimensional array containing the variable to be printed.

Description

Subroutine OUTPUT manages the printing of the standard output. The variables available for printing are listed and defined in Table 3-3 of Volume 2. The user-specified array IVOUT controls which variables are printed.

Each variable to be printed is stored, in turn, in the scratch array DUMMY, from the common block DUMMY1. The title printed with the variable is stored in the character array ATITLE. Subroutine PRTOUT is then called to execute the actual write statements.

Remarks

1. A warning message is printed if a non-existent output variable is requested. The printout will continue with the next requested output variable.
2. For output options 30, 31, 34, and 35, involving the pressure p, the value stored internally in the *Proteus* code is redefined as follows:

$$p = p\frac{\rho_r u_r^2}{p_r g_c}$$

   This is necessary because input and output values of p are nondimensionalized by the reference pressure $p_r = \rho_r \bar{R} T_r$, while internal to the code itself p is nondimensionalized by the normalizing pressure $p_n = \rho_r u_r^2$. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.
3. The definitions of k_l and k_t (IVOUT = 92 and 102) assume a constant turbulent Prandtl number is being specified in namelist TURB. If the input value of PRT ≤ 0, indicating the use of a variable turbulent Prandtl number, the printed values of k_l and k_t will be incorrect.

Subroutine OUTW (LEVEL)		
Called by	Calls	Purpose
MAIN		Compute and print parameters at boundaries.

Input

	CP	Specific heat c_p.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
	GC	Proportionality factor g_c in Newton's second law.
*	IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
*	IWOUT1, IWOUT2, IWOUT3	Flags specifying for which boundaries parameters are to be printed.
	LEVEL	Time level being printed.
	MU, KT	Effective coefficients of viscosity μ, and thermal conductivity k.
*	NOUT	Unit number for standard output.
	NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	P, T	Static pressure p and temperature T.
	PR	Reference pressure p_r.
	PRR	Reference Prandtl number Pr_r.
*	RER	Reference Reynolds number Re_r.
*	RHOR, UR	Reference density ρ_r, and velocity u_r.
	U, V, W	Velocities u, v, and w.
	X, Y, Z	Cartesian coordinates x, y, and z.
	XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
	ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

None.

Description

Subroutine OUTW computes and prints various parameters along the computational boundaries. The variables available for printing are listed and defined in Table 3-3 of Volume 2. The user-specified arrays IWOUT1, IWOUT2, and IWOUT3 specify at which boundaries parameters are printed, and whether normal derivatives are to be computed using two-point or three-point one-sided differencing.

The parameters printed are the Cartesian coordinates x, y, and z, the static pressure p, the skin friction coefficient c_f, the shear stress τ_w, the static temperature T, the heat transfer coefficient h, the heat flux q_w, and the Stanton number St. Note that some of these are meaningful only if the boundary is a solid wall.

The skin friction coefficient is defined as

$$c_f = \frac{\bar{\mu}\dfrac{\partial \bar{V}_t}{\partial \bar{n}}}{\dfrac{1}{2}\rho_r u_r^2} = \frac{2}{Re_r}\mu\frac{\partial V_t}{\partial n}$$

where the overbar denotes a dimensional quantity. In this equation $\partial V_t/\partial n$ represents the normal derivative of the tangential velocity, with the normal vector $\vec{n}$ directed into the flow field.

For a ξ boundary, the tangential velocity is

$$V_t = \sqrt{V_\eta^2 + V_\zeta^2}$$

where V_η and V_ζ are the velocities in the η and ζ directions. From the descriptions of subroutines BCV2 and BCV3,

$$V_\eta = \frac{x_\eta u + y_\eta v + z_\eta w}{\sqrt{x_\eta^2 + y_\eta^2 + z_\eta^2}}$$

$$V_\zeta = \frac{x_\zeta u + y_\zeta v + z_\zeta w}{\sqrt{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}}$$

Using the equations in Section 6.4 of Volume 1, $\partial V_t/\partial n$ for a ξ boundary is thus computed as

$$\frac{\partial V_t}{\partial n} = \pm\frac{1}{m}\left[\frac{\partial V_t}{\partial \xi}(\xi_x^2 + \xi_y^2 + \xi_z^2) + \frac{\partial V_t}{\partial \eta}(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z) + \frac{\partial V_t}{\partial \zeta}(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z)\right]$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}$$

The + sign is used at the $\xi = 0$ boundary, and the $-$ sign is used at the $\xi = 1$ boundary.

For an η boundary, the tangential velocity is

$$V_t = \sqrt{V_\xi^2 + V_\zeta^2}$$

From the description of subroutine BCV1,

$$V_\xi = \frac{x_\xi u + y_\xi v + z_\xi w}{\sqrt{x_\xi^2 + y_\xi^2 + z_\xi^2}}$$

Thus, for an η boundary,

$$\frac{\partial V_t}{\partial n} = \pm\frac{1}{m}\left[\frac{\partial V_t}{\partial \xi}(\eta_x\xi_x + \eta_y\xi_y + \eta_z\xi_z) + \frac{\partial V_t}{\partial \eta}(\eta_x^2 + \eta_y^2 + \eta_z^2) + \frac{\partial V_t}{\partial \zeta}(\eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z)\right]$$

where

$$m = \sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}$$

For an η boundary, the tangential velocity is

$$V_t = \sqrt{V_\xi^2 + V_\eta^2}$$

Thus, for a ζ boundary,

$$\frac{\partial V_t}{\partial n} = \pm \frac{1}{m} \left[\frac{\partial V_t}{\partial \xi} (\zeta_x \xi_x + \zeta_y \xi_y + \zeta_z \xi_z) + \frac{\partial V_t}{\partial \eta} (\zeta_x \eta_x + \zeta_y \eta_y + \zeta_z \eta_z) + \frac{\partial V_t}{\partial \zeta} (\zeta_x^2 + \zeta_y^2 + \zeta_z^2) \right]$$

where

$$m = \sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}$$

The shear stress τ_w is defined as

$$\tau_w = \mu \frac{\partial V_t}{\partial n}$$

τ_w is thus nondimensionalized by $\mu_r u_r / L_r$.

The heat flux q_w is defined as

$$q_w = -k \frac{\partial T}{\partial n}$$

where $\partial T / \partial n$ represents the normal derivative of the temperature. For a ξ boundary,

$$\frac{\partial T}{\partial n} = \pm \frac{1}{m} \left[\frac{\partial T}{\partial \xi} (\xi_x^2 + \xi_y^2 + \xi_z^2) + \frac{\partial T}{\partial \eta} (\xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z) + \frac{\partial T}{\partial \zeta} (\xi_x \zeta_x + \xi_y \zeta_y + \xi_z \zeta_z) \right]$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}$$

For an η boundary,

$$\frac{\partial T}{\partial n} = \pm \frac{1}{m} \left[\frac{\partial T}{\partial \xi} (\eta_x \xi_x + \eta_y \xi_y + \eta_z \xi_z) + \frac{\partial T}{\partial \eta} (\eta_x^2 + \eta_y^2 + \eta_z^2) + \frac{\partial T}{\partial \zeta} (\eta_x \zeta_x + \eta_y \zeta_y + \eta_z \zeta_z) \right]$$

where

$$m = \sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}$$

And for a ζ boundary,

$$\frac{\partial T}{\partial n} = \pm \frac{1}{m} \left[\frac{\partial T}{\partial \xi} (\zeta_x \xi_x + \zeta_y \xi_y + \zeta_z \xi_z) + \frac{\partial T}{\partial \eta} (\zeta_x \eta_x + \zeta_y \eta_y + \zeta_z \eta_z) + \frac{\partial T}{\partial \zeta} (\zeta_x^2 + \zeta_y^2 + \zeta_z^2) \right]$$

where

$$m = \sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}$$

q_w is thus nondimensionalized by $k_r T_r / L_r$.

The heat transfer coefficient h is defined as

$$h = \frac{q_w}{T - 1} = \frac{-k \frac{\partial T}{\partial n}}{T - 1}$$

This is the nondimensional form of the equation

$$\bar{h} = \frac{\bar{q}_w}{\bar{T} - T_r} = \frac{-\bar{k}\dfrac{\partial \bar{T}}{\partial \bar{n}}}{\bar{T} - T_r}$$

h is thus nondimensionalized by k_r/L_r.

The Stanton number St is defined as

$$St = \frac{\bar{h}}{\rho_r u_r \bar{C}_p} = \frac{h}{c_p}\frac{1}{Re_r Pr_r}$$

Subroutine PAK (IDIR,NOLD1,NOLD2,NOLD3)		
Called by	Calls	Purpose
GEOM	CUBIC ROBTS	Manage packing and/or interpolation of grid points.

Input

IDIR	Direction flag; 1 if grid points are being redistributed in the ξ direction, 2 if in the η direction, 3 if in the ζ direction.
* IPACK	Flags for grid packing option.
NOLD1, NOLD2, NOLD3	Number of grid points in the ξ, η, and ζ directions in the original grid.
* NOUT	Unit number for standard output.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
* SQ	An array specifying the location and amount of packing.
X, Y, Z	Cartesian coordinates x, y, and z in the old grid.

Output

X, Y, Z	Cartesian coordinates x, y, and z in the new grid.

Description

Subroutine PAK manages the redistribution of the user-specified points in the computational coordinate system. It is called whenever grid packing is used. It is also called when interpolation is necessary because the computational coordinates are specified by reading them from a separate file (the NGEOM = 10 option in subroutine GEOM), and the number of points in the file is different from the number of points to be used in the calculation. PAK is called once for each direction in which points are being redistributed.

The steps involved in subroutine PAK are described below. For clarity, this discussion assumes IDIR = 1 (i.e., we are redistributing points in the ξ direction.) An exactly analogous procedure is used for IDIR = 2 and 3.

1. Set NNEW and NOLD equal to the index limits in the ξ direction for the new and old grids. Also set NOPP1 and NOPP2 equal to the index limits in the η and ζ directions for the old grid.
2. Get $(a_P)_i$, the normalized physical arc length along a coordinate line in the ξ direction, from the beginning of the line to each grid point in the new grid. The normalizing distance is the total arc length of the line, and thus these arc lengths apply to any coordinate line in the ξ direction. If the points are not being packed in the ξ direction, but only interpolated, then

$$(a_P)_i = \frac{i-1}{\mathrm{NNEW}-1}$$

   for $i = 1$ to NNEW. In the new grid, the points will thus be evenly distributed in physical space along each coordinate line in the ξ direction. If the grid points are being packed in the ξ direction, subroutine ROBTS is called to compute $(a_P)_i$ from the packing parameters specified by the user.
3. Begin double loop from IOPP1 = 1 to NOPP1 and from IOPP2 = 1 to NOPP2. This double loop thus runs over the points in the η and ζ directions in the old grid. We will be redistributing points in the ξ direction for each η and ζ value in the old grid.

4. Get $(a_{UP})_i$, the normalized physical arc length along a coordinate line in the ξ direction, from the beginning of the line to each grid point in the old grid. These values are found by first computing the non-normalized arc lengths, as follows:

$$(a_{UP})_1 = 0$$

$$(a_{UP})_i = (a_{UP})_{i-1} + \sqrt{(x_{i,j,k} - x_{i-1,j,k})^2 + (y_{i,j,k} - y_{i-1,j,k})^2}$$

for $i = 2$ to NOLD1. These values are normalized by setting

$$(a_{UP})_i = \frac{(a_{UP})_i}{(a_{UP})_{\mathrm{NOLD1}}}$$

for $i = 1$ to NOLD1. To eliminate any problems with roundoff error, $(a_{UP})_{\mathrm{NOLD1}}$ is explicitly set equal to 1.

5. Given x and a_{UP} for the old grid, and a_P for the new grid, call CUBIC to interpolate for x in the new grid. Similarly interpolate for y and z.

6. Redefine the Fortran variables X, Y, and Z as the x, y, and z coordinates in the new grid.

7. End of double loop over the points in the η and ζ directions in the old grid.

Remarks

1. In the Fortran code, the comments sometimes refer to the "packing" direction. This terminology actually means the direction in which grid points are being redistributed, even if they are not being packed but only interpolated. Similarly, references to the "packed" and "unpacked" grid actually mean the new and old grids.

2. An error message is generated and execution is stopped if an invalid grid packing option is requested.

Subroutine PERIOD		
Called by	Calls	Purpose
EXEC EXECT		Define extra line of data for use in computing coefficients for spatially periodic boundary conditions.

Input

CP, CV	Specific heats c_p and c_v at time level n.
E, EL	Turbulent dissipation rate ε at time levels n and $n-1$.
ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
KE, KEL	Turbulent kinetic energy k at time levels n and $n-1$.
MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k.
MUT, MUTL	Turbulent viscosity μ_t at time levels n and $n-1$.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, or N_1+1, N_2+1, and N_3+1 for spatially periodic boundary conditions, in ξ, η, and ζ, respectively.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
P, T	Static pressure p and temperature T at time level n.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T from previous ADI sweep.
TL	Static temperature T from previous ADI sweep.
XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.
ZETAX, ZETAY, ZETAZ, ZETAT	Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

All of the flow and metric-related input parameters listed above, at $i = N_1 + 1$ for periodic boundary conditions in the ξ direction, at $j = N_2 + 1$ for periodic boundary conditions in the η direction, and at $k = N_3 + 1$ for periodic boundary conditions in the ζ direction.

Description

Subroutine PERIOD adds, in effect, an additional set of points at $i = N_1 + 1$ for periodic boundary conditions in the ξ direction, at $j = N_2 + 1$ for periodic boundary conditions in the η direction, and at $k = N_3 + 1$ for periodic boundary conditions in the ζ direction. This allows us to use central differencing in the periodic direction, at $i = N_1$, $j = N_2$, and/or $k = N_3$, computing the coefficient submatrices and source term subvector in the same way as at the interior points.[29]

[29] See Section 7.2.2 of Volume 1 for details on the solution procedure for spatially periodic boundary conditions.

For periodic boundary conditions in the ξ direction, the extra points are added by setting

$$f_{N_1+1,j,k} = f_{2,j,k}$$

where $j = 1$ to N_2 and $k = 1$ to N_3, and f represents one of the flow variables or metrics. Similarly, extra points are added at $(i, N_2 + 1, k)$ for periodic boundary conditions in the η direction, and at $(i, j, N_3 + 1)$ for periodic boundary conditions in the ζ direction.

Subroutine PLOT (LEVEL)		
Called by	Calls	Purpose
MAIN		Write files for post-processing by CONTOUR or PLOT3D plotting programs.

Input

	CP, CV	Specific heats c_p and c_v.
	ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
*	GAMR	Reference ratio of specific heats, γ_r.
	GC	Proportionality factor g_c in Newton's second law.
*	IPLOT	Flag specifying type of plot file to be written.
	LEVEL	Time level to be written into the file (0 for initialization,)
*	LR, UR, RHOR, TR	Reference length L_r, velocity u_r, density ρ_r, and temperature T_r.
*	MACHR	Reference Mach number M_r.
*	NOUT	Unit number for standard output.
*	NPLOT	Unit number for writing CONTOUR file, or PLOT3D Q file.
*	NPLOTX	Unit number for writing PLOT3D XYZ file.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	P, T	Static pressure p and temperature T.
	PR	Reference pressure p_r.
*	RER	Reference Reynolds number Re_r.
*	RG	Dimensional gas constant $\bar{R}$.
	RGAS	Dimensionless gas constant R.
	RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T.
	TAU	Current time value τ.
*	TITLE	Case title.
	X, Y, Z	Cartesian coordinates x, y, and z.
	XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
	ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

None.

Description

Subroutine PLOT writes a file or files, commonly called plot files, for post-processing by the CONTOUR or PLOT3D plotting programs. The type of files written is controlled by the user-specified parameter IPLOT. The format and contents of the different types of plot files are described in detail in Section 4.2 of Volume 2. They are therefore described only briefly here.

CONTOUR Plot File (IPLOT = 1)

If IPLOT = 1, a CONTOUR plot file is written with the title and reference conditions included at each time level. The value of n is written into the header for each time level, but τ, the time itself, is not written into the file. No initialization step is necessary.

PLOT3D/WHOLE Plot Files (IPLOT = 2)

If IPLOT = 2, XYZ and Q files are written in PLOT3D/WHOLE format. The XYZ file is written only during the initialization step. The Q file is written at each time level requested by the user. The Q file will thus consist of multiple sets of data, each containing the computed results at a single time level. The time $\tau_{1,1,1}$ is written into the header for each set of data in the Q file.

PLOT3D/PLANES Plot Files (IPLOT = 3)

If IPLOT = 3, XYZ and Q files are written in PLOT3D/PLANES format. The XYZ file is written only during the initialization step. The Q file is written at each time level requested by the user. The Q file will thus consist of multiple sets of data, each containing the computed results at a single time level. The time $\tau_{1,1,1}$ is written into the header for each set of data in the Q file.

Remarks

1. In defining the pressure to be written into the CONTOUR plot file, the value stored internally in the *Proteus* code is redefined as follows:

$$p = p\,\frac{\rho_r u_r^2}{p_r g_c}$$

   This is necessary because input and output values of p are nondimensionalized by the reference pressure $p_r = \rho_r \bar{R} T_r$, while internal to the code itself p is nondimensionalized by the normalizing pressure $p_n = \rho_r u_r^2$. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.

2. The current version of PLOT3D does not work for multiple time levels, although future versions might. Thus the IPLOT = 2 and 3 options, while containing multiple time levels, cannot easily be used to create plots showing the time development of the flow.

3. Note that the time $\tau_{1,1,1}$ written into the Q file header with the IPLOT = 2 and 3 options is the time at the point $\xi = \eta = \zeta = 0$. If the input variable IDTAU = 5 or 6, τ will vary in space and therefore $\tau_{i,j,k} \neq \tau_{1,1,1}$.

4. PLOT3D assumes that velocity is nondimensionalized by the reference speed of sound $a_r = (\gamma_r \bar{R} T_r)^{1/2}$, and that energy is nondimensionalized by $\rho_r a_r^2$. In *Proteus* these variables are nondimensionalized by u_r and $\rho_r u_r^2$. That is why the reference Mach number M_r appears in the definitions of the Q variables written into the plot file.

6. An error message is generated and execution is stopped if an illegal plot file option is requested.

Subroutine PRODCT		
Called by	Calls	Purpose
KEINIT TURBCH		Compute production term for the k-ε turbulence model.

Input

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
KE	Turbulent kinetic energy k at time level n.
MUT	Turbulent viscosity μ_t at time level n.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
* RER	Reference Reynolds number Re_r.
RHO, U, V, W	Density ρ, and velocities u, v, and w at time level n.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VORT	Production rate of turbulent kinetic energy.

Description

Subroutine PRODCT computes the turbulent kinetic energy production rate using

$$P_k = \frac{\mu_t}{Re_r} P_1 - \frac{2}{3} \rho k P_2$$

where

$$P_1 = 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right] - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial u}{\partial z} + \frac{\partial v}{\partial x} + \frac{\partial v}{\partial z} + \frac{\partial w}{\partial x} + \frac{\partial w}{\partial y}\right)^2$$

$$P_2 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

To evaluate the spatial derivatives, the centered difference formulas presented in Section 5.0 of Volume 1 are used at interior points, and second-order one-sided difference formulas are used at boundary points.

Remarks

1. To save storage space, this subroutine uses the Fortran variable VORT to store the turbulent kinetic energy production rate. Care must be taken when this subroutine is used together with subroutine VORTEX.

Subroutine PRTHST		
Called by	Calls	Purpose
MAIN		Print convergence history.

Input

*	ICHECK	Convergence checking interval.
*	IREST	Flag for reading/writing restart file.
	IT	Last computed time step number n.
	ITBEG	The time level n at the beginning of a run.
	NC, NXM, NYM, NZM, NEN	Array indices associated with the continuity, x-momentum, y-momentum, z-momentum, and energy equations.
	NEQ	Number of coupled equations being solved, N_{eq}.
*	NHIST	Unit number for convergence history file.
*	NHMAX	Maximum number of time levels allowed in the printout of the convergence history file (not counting the first two, which are always printed.)
*	NOUT	Unit number for standard output.

Output

None.

Description

Subroutine PRTHST prints the convergence history as part of the standard output. The information is obtained fro n the unformatted convergence history file written in subroutine RESID. The arameters printed are described in Section 4.1.6 of Volume 2, and the unformatted convergence history file is described in Section 4.3 of Volume 2. To avoid undesirably long tables, the convergence parameters are printed at an interval that limits the printout to NHMAX time levels. As described in Section 4.1.6 of Volume 2, however, they are always printed at the first two time levels.

Subroutine PRTOUT (ATITLE,LEVEL,AVAR)		
Called by	Calls	Purpose
OUTPUT		Print output.

Input

ATITLE	A 20-character title for variable being printed.
AVAR	A three-dimensional array containing the variable to be printed.
DTAU	Time step $\Delta\tau$.
* IDTAU	Flag for time step selection method.
* IPRT1A, IPRT2A, IPRT3A	Indices for printout in the ξ, η, and ζ directions.
LEVEL	Time level to be printed.
* LR, UR	Reference length L_r and velocity u_r.
* NOUT	Unit number for standard output.
NPRT1, NPRT2, NPRT3	Total number of indices for printout in the ξ, η, and ζ directions.
TAU	Current time value τ.

Output

None.

Description

Subroutine PRTOUT performs the actual printing of the standard output file. It prints the variable AVAR, with the title ATITLE. The output is printed in blocks, with each block corresponding to a ζ location. Within each ζ block, the output is printed in columns running in the η direction. The rows run in the ξ direction. If the results at every grid point are printed, there will be a total of N_3 blocks, each block with N_1 columns, and each column with N_2 rows. Within each ζ block, the columns are grouped in super-rows of up to 10 columns each.

The steps involved are as follows:

1. Set the total number of blocks, columns, and rows per super-row.
2. Redefine AVAR, the input array containing the variable to be printed, including only the elements requested.
3. Determine the number of super-rows. If NCOL is not exactly divisible by 10, the last super-row in each block will have less than 10 columns.
4. Begin loop over the number of ζ blocks.
5. Print the title for the variable, and the ζ index. If the time step is constant in space, the dimensional time t and time step Δt are printed with the title.
6. Begin loop over the number of super-rows.
7. Set NC1 and NC2 equal to the number of the first and last column in this super-row. (I.e, for the first super-row NC1 and NC2 will be 1 and 10, for the second they will be 11 and 20, etc. For the last super-row, NC2 will be NCOL.)
8. Print the heading for the super-row, labeling each column with the proper ξ index.
9. Print the super-row itself, labeling each row with the proper η index.

10. End of loop over the number of super-rows.

11. End of loop over the number of ζ blocks.

Subroutine RESID (IAVR)		
Called by	Calls	Purpose
EXEC	ISAMAX SASUM SNRM2	Compute residuals and write convergence history file.

Input

	CHGAVG	Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps, ΔQ_{avg}.
	CHGMAX	Maximum change in absolute value of the dependent variables over previous time step (or NITAVG − 1 time steps if ICTEST = 2), ΔQ_{max}.
	DTAU	Time step $\Delta\tau$.
	DUMMY	A three-dimensional scratch array.
*	EPS	Convergence level to be reached, ε.
	IAVR	Flag specifying whether residual is computed without or with the artificial viscosity terms; 1 for without, 2 for with.
*	IAV2E, IAV4E	Flags for second- and fourth-order explicit artificial viscosity.
*	ICHECK	Convergence checking interval.
*	ICTEST	Flag for convergence criteria to be used.
*	IDTAU	Flag for time step selection method.
*	IHSTAG	Flag for constant stagnation enthalpy option.
	IT	Current time step number n.
	ITBEG	The time level n at the beginning of a run.
	I3	Grid index k in the ζ direction.
*	LR, UR	Reference length L_r and velocity u_r.
	LRMAX	Grid indices i, j, and k, in the ξ, η, and ζ directions, corresponding to the location of RESMAX.
	NEQ	Number of coupled equations being solved, N_{eq}.
*	NHIST	Unit number for convergence history file.
*	NITAVG	Number of time steps in moving average convergence test.
	NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	N1P, N2P	Parameters specifying the dimension sizes in the ξ and η directions.
	RESAVG	The sum of the absolute values of the residual through the ζ index I3 − 1.
	RESL2	The sum of the squares of the residual through the ζ index I3 − 1.

RESMAX — The maximum absolute value of the residual, R_{max}, through the ζ index I3 − 1.

S — Source term subvector S for first ADI sweep.

TAU — Current time value τ.

Output

LRMAX — Grid indices i, j, and k, in the ξ, η, and ζ directions, corresponding to the location of RESMAX.

RESAVG — The sum of the absolute values of the residual through the ζ index I3, or, if I3 = NPT3 − 1, the average absolute value of the residual, R_{avg}.

RESL2 — The sum of the squares of the residual through the ζ index I3, or, if I3 = NPT3 − 1, the L_2 norm of the residual, R_{L_2}.

RESMAX — The maximum absolute value of the residual, R_{max}, through the ζ index I3.

Description

Subroutine RESID computes various measures of the residual, and writes the convergence history file.

For problems without artificial viscosity, the steady-state form of the governing partial differential equations can be written as

$$0 = -\frac{\partial \hat{\mathbf{E}}}{\partial \xi} - \frac{\partial \hat{\mathbf{F}}}{\partial \eta} - \frac{\partial \hat{\mathbf{G}}}{\partial \zeta} + \frac{\partial \hat{\mathbf{E}}_V}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}_V}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}_V}{\partial \zeta}$$

The residual is defined as the number resulting from evaluating the right hand side of the above equation. For first-order time differencing, this is simply the source term for the first ADI sweep, divided by the time step $\Delta\tau$.[30] The residual at a specific grid point and time level is thus

$$R^n_{i,j,k} = S^n_{i,j,k} / (\Delta\tau)^n_{i,j,k}$$

where S is the source term for the first ADI sweep. Separate residuals are computed for each governing equation.

Adding artificial viscosity, however, changes the governing equations. With artificial viscosity, the difference equations actually correspond to the following differential equations at steady state.[31]

$$\begin{aligned} 0 = & -\frac{\partial \hat{\mathbf{E}}}{\partial \xi} - \frac{\partial \hat{\mathbf{F}}}{\partial \eta} - \frac{\partial \hat{\mathbf{G}}}{\partial \zeta} + \frac{\partial \hat{\mathbf{E}}_V}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}_V}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}_V}{\partial \zeta} \\ & + \frac{\varepsilon_E^{(2)}}{J} \left[(\Delta\xi)^2 \frac{\partial^2 (J\hat{\mathbf{Q}})}{\partial \xi^2} + (\Delta\eta)^2 \frac{\partial^2 (J\hat{\mathbf{Q}})}{\partial \eta^2} + (\Delta\zeta)^2 \frac{\partial^2 (J\hat{\mathbf{Q}})}{\partial \zeta^2} \right] \\ & - \frac{\varepsilon_E^{(4)}}{J} \left[(\Delta\xi)^4 \frac{\partial^4 (J\hat{\mathbf{Q}})}{\partial \xi^4} + (\Delta\eta)^4 \frac{\partial^4 (J\hat{\mathbf{Q}})}{\partial \eta^4} + (\Delta\zeta)^4 \frac{\partial^4 (J\hat{\mathbf{Q}})}{\partial \zeta^4} \right] \end{aligned}$$

---

30 See equation (7.5a) in Volume 1. For first-order time differencing, $\theta_2 = \theta_3 = 0$.

31 These equations represent the use of the constant coefficient artificial viscosity model. The nonlinear coefficient model is more complicated, but the same principle applies.

For cases run with artificial viscosity, therefore, the residual should include the explicit artificial viscosity terms. The implicit terms do not appear, since they difference $\Delta\hat{Q}$, and in the steady form of the equations $\Delta\hat{Q} = 0$. Since the explicit artificial viscosity terms are added to the source term for the first ADI sweep, they are automatically included in the residual.

Three measures of the residual are computed for each governing equation - the L_2 norm of the residual, the average absolute value of the residual, and the maximum absolute value of the residual. In addition, the (ξ,η,ζ) indices corresponding to the location of the maximum residual are saved. The L_2 norm of the residual is defined as

$$R_{L_2} = \left(\sum (R_{i,j,k})^2 \right)^{1/2}$$

In computing the residuals, the summations, maximums, and averages are over all interior grid points, plus points on spatially periodic boundaries. RESID is called from inside a loop in the ζ direction. The calculation of the residual is thus not complete until the last time through this loop, when the ζ index $I3 = NPT3 - 1$.

For cases run with artificial viscosity, subroutine RESID is called from EXEC both before and after the artificial viscosity terms have been added to the equations. The residuals are thus computed both with and without the artificial viscosity terms. This may provide some estimate of the overall error in the solution introduced by the artificial viscosity. Convergence is determined by the residuals with the artificial viscosity terms included.

In addition to computing the residuals, subroutine RESID writes the convergence history file. The contents and format of this file are described in detail in Section 4.3 of Volume 2.

Remarks

1. The Cray BLAS routines SNRM2 and SASUM are used in computing the L_2 norm of the residual and the average absolute value of the residual, respectively. The Cray search routine ISAMAX is used in computing the maximum absolute value of the residual.
2. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the residual at each grid point.

Subroutine REST (IOPT)		
Called by	Calls	Purpose
INITC MAIN	METS	Read and/or write restart file.

Input When Reading the Restart File

* GAMR	Reference ratio of specific heats, γ_r.
* HSTAG	Stagnation enthalpy h_T used with constant stagnation enthalpy option.
* IHSTAG	Flag for constant stagnation enthalpy option.
IOPT	Flag specifying I/O operation; 1 to read, 2 to write.
* ITURB	Flag for turbulent flow option.
* NRQIN	Unit number for reading the restart flow field.
* NRXIN	Unit number for reading the restart computational mesh.
RGAS	Dimensionless gas constant R.

Input When Writing the Restart File

E, KE	Turbulent dissipation rate ε and kinetic energy k at time level $n+1$.
EL, KEL	Turbulent dissipation rate ε and kinetic energy k at time level n.
IOPT	Flag specifying I/O operation; 1 to read, 2 to write.
IT	Current time step number n.
* ITURB	Flag for turbulent flow option.
* MACHR	Reference Mach number M_r.
* NRQOUT	Unit number for writing the restart flow field.
* NRXOUT	Unit number for writing the restart computational mesh.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
* RER	Reference Reynolds number Re_r.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level $n+1$.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
X, Y, Z	Cartesian coordinates x, y, and z.

Output When Reading the Restart File

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
E, KE	Turbulent dissipation rate ε and kinetic energy k at time level ITBEG.
EL, KEL	Turbulent dissipation rate ε and kinetic energy k at time level ITBEG − 1.

ITBEG	The time level n at the beginning of the new run.
MACHR	Reference Mach number M_r.
N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
RER	Reference Reynolds number Re_r.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level ITBEG.
RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at time level ITBEG $-$ 1.
T, TL	Static temperature T at time levels ITBEG and ITBEG $-$ 1.
X, Y, Z	Cartesian coordinates x, y, and z.

Output When Writing the Restart File

None.

Description

Subroutine REST reads and/or writes the restart files. Restarting a calculation requires two unformatted files - one containing the computational mesh and one containing the flow field.

If subroutine REST is being used to read the restart files, the computational mesh is first read from unit NRXIN. The grid increments $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$ are then set, and subroutine METS is called to compute the metric coefficients and the Jacobian of the grid transformation.

The flow field file is read next, from unit NRQIN. It normally contains the results at the last two time levels that were computed during the previous run. If only one level is present in the file, however, the results at level $n-1$ are set equal to those at level n. If the previous run used the two-equation turbulence model, the turbulence variables are also read from the file. The beginning time level for the time marching loop is set equal to the level stored in the restart file. The flow field variables in the restart file are the conservation variables **Q**, nondimensionalized as in the plotting program PLOT3D.[32] They therefore must be converted into the primitive variables used in *Proteus*. The temperature is then computed from the perfect gas equation of state, with c_p and c_v defined using the input reference conditions.

When writing the restart files, the file containing the computational mesh is written onto unit NRXOUT. The primitive flow variables are then redefined as conservation variables and nondimensionalized as in PLOT3D. They are then written onto unit NRQOUT. If the current run used the two-equation turbulence model, the turbulence variables are also written into the file.

Remarks

1. If, in the input namelist RSTRT, NRXOUT and NRQOUT are set equal to NRXIN and NRQIN, respectively, the output restart files will overwrite the input restart files.
2. Except for the turbulence variables and the variables at time level $n-1$, the restart files have the same format as the XYZ and Q files created using the IPLOT = 3 option. These restart files can thus also be used as XYZ and Q files for the PLOT3D plotting program. The turbulence variables and the variables at time level $n-1$ will not be read by PLOT3D.
3. The temperature T is computed using the equation of state, which contains a specific heat coefficient (either c_p or c_v, depending on whether the stagnation enthalpy is assumed constant or not.) In subroutine REST, a constant value of specific heat is used, consistent with the reference temperature T_r. If the user specified constant specific heat (i.e., a value for γ_r was read in), this is not a problem. However, if the temperature-dependent specific heat option is being used (i.e., a value for γ_r was not

[32] See Sections 4.2.3 and 4.4 of Volume 2.

read in), the equation of state and the empirical equation for specific heat are coupled. For this reason, in INITC (the routine that calls REST), T is recomputed by calling EQSTAT after the specific heats have been computed in FTEMP. Ideally, this coupling would be handled by iteration between FTEMP and EQSTAT. This is not currently done in *Proteus*, however.

Subroutine ROBTS (NP,A,B,XP)		
Called by	Calls	Purpose
PAK		Pack points along a line using Roberts transformation.

Input

A — Parameter α in Roberts transformation formula specifying location of packing: 0.0 to pack near XP = 1 only, 1.0 to pack near XP = 0 only, and 0.5 to pack equally at XP = 0 and 1.0.

B — Parameter β in Roberts transformation formula specifying amount of packing. A value approaching 1.0 from above gives denser packing.

NP — Number of grid points along the line.

Output

XP — Coordinates of packed grid points along the line.

Description

Subroutine ROBTS packs points along a line of length one using a transformation due to Roberts (1971). The basic transformation is given by

$$x_P = \frac{(\beta + 2\alpha)\beta_r^{\beta_x} - \beta + 2\alpha}{(2\alpha + 1)(1 + \beta_r^{\beta_x})}$$

where

$$\beta_r = \frac{\beta + 1}{\beta - 1}$$

$$\beta_x = \frac{x_{UP} - \alpha}{1 - \alpha}$$

and x_P and x_{UP} are the packed and unpacked (i.e., evenly spaced) coordinates along the line. The parameter α determines the packing location. For $\alpha = 0$, the points will be packed only near $x_P = 1$, and for $\alpha = 1/2$ the points will be packed equally near $x_P = 0$ and $x_P = 1$. The packing parameter β determines the amount of packing. It is a number greater than 1, but generally 1.1 or below. The closer β is to 1, the tighter the packing will be.

It may seem logical to set $\alpha = 1$ to pack points near $x_P = 0$. With the basic transformation, however, this doesn't work. In *Proteus* we get around this problem by replacing α in the above transformation with α_w, where $\alpha_w = \alpha$ if $\alpha = 0$ or $1/2$, and $\alpha_w = 0$ if $\alpha = 1$. If $\alpha = 0$ or $1/2$, no further action is necessary. If $\alpha = 1$, however, we must invert the resulting x_P values and re-order the indices. I.e., for $i = 1$ to NP, we set

$$(x_{PI})_i = 1 - (x_P)_i$$

After this operation, the array x_{PI} will run from 1 to 0, packed near 1. To re-order the indices, for $i = 1$ to NP we set

$$(x_P)_{NP - i + 1} = (x_{PI})_i$$

After this operation, x_P will run from 0 to 1, packed near 0.

Finally, to ensure round-off error doesn't affect the endpoint values, we set $(x_P)_1 = 0$ and $(x_P)_{NP} = 1$.

Remarks

1. The namelist input variable SQ(IDIR,1), which is used to specify the packing location in direction IDIR, is actually equal to $1 - \alpha$. Therefore, setting SQ(IDIR,1) = 0 results in packing near the ξ, η, or $\zeta = 0$ boundary, and SQ(IDIR,1) = 1 results in packing near the ξ, η, or $\zeta = 1$ boundary.

Function SASUM (N,V,INC)		
Called by	**Calls**	**Purpose**
RESID		Compute the sum of the absolute values of the elements of a vector.

Input

N — Number of elements in the vector to be summed.

V — Vector to be summed.

INC — Skip distance between elements of V. For contiguous elements, INC = 1.

Output

SASUM — Sum of the absolute values of the elements of V.

Description

Function SASUM computes the sum of the absolute values of the elements of a vector. For a one-dimensional vector, the use of SASUM is straightforward. For example,

$$\texttt{sasum(np,v,1)} = \sum_{i=1}^{np} V_i$$

A starting location can be specified, as in

$$\texttt{sasum(np-4,v(5),1)} = \sum_{i=5}^{np} V_i$$

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2 by NDIM3, then

$$\texttt{sasum(ndim1*ndim2*ndim3,a,1)} = \sum_{i=1}^{ndim1} \sum_{j=1}^{ndim2} \sum_{k=1}^{ndim3} A_{i,j,k}$$

One dimension at a time can also be summed. For example,

$$\texttt{sasum(ndim1,a(1,5,2),1)} = \sum_{i=1}^{ndim1} A_{i,5,2}$$

Similarly, by specifying a skip increment,

$$\texttt{sasum(ndim2,a(5,1,2),ndim1)} = \sum_{j=1}^{ndim2} A_{5,j,2}$$

Remarks

1. SASUM is a Cray BLAS (Basic Linear Algebra Subprograms) routine (Cray Research, Inc., 1989b).

Subroutine SGEFA (A,LDA,N,IPVT,INFO)		
Called by	Calls	Purpose
BCELIM BVUP	ISAMAX	Factor a matrix using Gaussian elimination.

Input

A — An array containing the matrix **A** to be factored, dimensioned as A(LDA,N).

LDA — The leading dimension of the array A.

N — The order of the matrix **A**.

Output

A — An upper triangular matrix and the multipliers which were used to obtain it. The factorization can be written as $\mathbf{A} = \mathbf{LU}$, where **L** is a product of permutation and unit lower triangular matrices, and **U** is upper triangular.

IPVT — A vector of length N containing pivot indices.

INFO — An error flag: 0 for normal operation, k if $U_{kk} = 0$.

Description

Subroutine SGEFA is used in combination with subroutine SGESL to solve the matrix equation $\mathbf{Ax} = \mathbf{B}$. If the Fortran arrays A and B represent **A** and **B**, where **A** is a square N by N matrix and **B** is a matrix (or vector) with NCOL columns, and if the leading dimension of the Fortran array A is LDA, then the Fortran sequence

```
      call sgefa (a,lda,n,ipvt,info)
      do 10 j = 1,ncol
      call sgesl (a,lda,n,ipvt,b(1,j),0)
10    continue
```

computes $\mathbf{A}^{-1}\mathbf{B}$, storing the result in **B**.

Remarks

1. SGEFA is a Cray LINPACK routine (Cray Research, Inc., 1989b; Dongarra, Moler, Bunch, and Stewart, 1979).

Subroutine SGESL (A,LDA,N,IPVT,B,JOB)		
Called by	Calls	Purpose
BCELIM BVUP		Solve the matrix equation $\mathbf{Ax} = \mathbf{B}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{B}$ using the factors computed by SGEFA.

Input

A	The two-dimensional output array A from SGEFA containing the factorization of matrix **A**.
B	The right-hand side vector **B**.
IPVT	The output array IPVT of pivot indices from SGEFA.
JOB	Flag specifying type of matrix equation: 0 to solve $\mathbf{Ax} = \mathbf{B}$; nonzero to solve $\mathbf{A}^T\mathbf{x} = \mathbf{B}$.
LDA	The leading dimension of the array A.
N	The order of the matrix A.

Output

B	The solution vector **x**.

Description

Subroutine SGESL is used in combination with subroutine SGEFA to solve the matrix equation $\mathbf{Ax} = \mathbf{B}$. See the description of subroutine SGEFA for details.

Remarks

1. SGESL is a Cray LINPACK routine (Cray Research, Inc., 1989b; Dongarra, Moler, Bunch, and Stewart, 1979).

Function SNRM2 (N,V,INC)		
Called by	Calls	Purpose
RESID		Compute the L_2 norm of a vector.

Input

N	The number of elements in the vector V.
V	The vector whose norm is to be computed.
INC	Skip distance between elements of V. For contiguous elements, INC = 1.

Output

SNRM2	The L_2 norm of the vector V.

Description

Function SNRM2 computes the L_2 norm of a vector. For a one-dimensional vector, the use of SNRM2 is straightforward. For example,

$$\texttt{snrm2(np,v,1)} = \left(\sum_{i=1}^{np} V_i^2 \right)^{1/2}$$

A starting location can be specified, as in

$$\texttt{snrm2(np-4,v(5),1)} = \left(\sum_{i=5}^{np} V_i^2 \right)^{1/2}$$

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2 by NDIM3, then

$$\texttt{snrm2(ndim1*ndim2*ndim3,a,1)} = \left(\sum_{i=1}^{ndim1} \sum_{j=1}^{ndim2} \sum_{k=1}^{ndim3} A_{i,j,k}^2 \right)^{1/2}$$

One dimension at a time can also be summed. For example,

$$\texttt{snrm2(ndim1,a(1,5,2),1)} = \left(\sum_{i=1}^{ndim1} A_{i,5,2}^2 \right)^{1/2}$$

Similarly, by specifying a skip increment,

$$\texttt{snrm2(ndim2,a(5,1,2),ndim1)} = \left(\sum_{j=1}^{\text{ndim2}} A_{5,j,2}^2 \right)^{1/2}$$

Remarks

1. SNRM2 is a Cray BLAS (Basic Linear Algebra Subprograms) routine (Cray Research, Inc., 1989b).

Subroutine SWDOWN		
Called by	Calls	Purpose
EXECT		Compute coefficients and source terms, and solve the k-ε equations for the downward LU sweep.

Input

* CMUR	Constant C_{μ_r} in formula for C_μ.
* CONE	Constant C_1 in the production term of the ε equation.
* CTHREE	Constant C_3 in formula for C_μ.
* CTWOR	Constant C_{2_r} in formula for C_2.
DTAU	Time step $\Delta\tau$.
DUMMY	Distance to the nearest solid wall.
DW	Dependent variable subvector $\Delta\hat{W}^*$ from upward LU sweep.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
E	Turbulent dissipation rate ε at time level n.
ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
KE	Turbulent kinetic energy k at time level n.
MU	Laminar viscosity μ_l at time level n.
MUT	Turbulent viscosity μ_t at time level n.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
* RER	Reference Reynolds number Re_r.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.
* SIGE, SIGK	Constants σ_ε and σ_k used in the diffusion term of the ε equation and k equations, respectively.
* TFACT	Factor used in computing the k-ε time step.
* THKE	Parameters θ_1 and θ_2 determining type of time differencing for the k-ε equations.
VORT	Production rate of turbulent kinetic energy.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
YPLUSD	Nondimensional distance y^+ from the nearest solid wall.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

DW	Dependent variable subvector $\Delta\hat{W}^n$ from downward LU sweep.

Description

Subroutine SWDOWN performs the downward LU sweep to solve for the final values of the unknown vector $\Delta\hat{\mathbf{W}}^n$ in the k-ε equations. The equation used for the downward sweep is

$$\left\{\mathbf{I} + \frac{\theta_1 \Delta\tau}{1+\theta_2}\left[\delta_\xi^+ A^- + \delta_\eta^+ C^- + \delta_\zeta^+ E^- - (\delta_\xi B)^+ - (\delta_\eta D)^+ - (\delta_\zeta F)^+ - (M+N)\right]\right\}^n \Delta\hat{\mathbf{W}}^n = \Delta\hat{\mathbf{W}}^*$$

The Jacobian coefficient matrices B, D, and F are made up of terms of the form $f\partial g/\partial\xi$, $f\partial g/\partial\eta$, and $f\partial g/\partial\zeta$, respectively. The terms $(\delta_\xi B)^+$, $(\delta_\eta D)^+$, and $(\delta_\zeta F)^+$ are the forward difference parts of the central differences $\delta_\xi B$, $\delta_\eta D$, and $\delta_\zeta F$, respectively. Thus

$$\delta_\xi B = \delta_\xi\left(f\frac{\partial g}{\partial\xi}\right) = \frac{(f_{i+1}+f_i)(g_{i+1}-g_i) - (f_i + f_{i-1})(g_i - g_{i-1})}{2(\Delta\xi)^2}$$

$$(\delta_\xi B)^+ = \frac{(f_{i+1}+f_i)(g_{i+1}-g_i)}{2(\Delta\xi)^2}$$

Analogous expressions can be derived for D and F. Expanding the difference terms in the downward sweep equation, we thus get

$$\left\{\mathbf{I} + \frac{\theta_1\Delta\tau}{1+\theta_2}\left[\frac{A^-_{i+1} - A^-_i}{\Delta\xi} + \frac{C^-_{j+1} - C^-_j}{\Delta\eta} + \frac{E^-_{k+1} - E^-_k}{\Delta\zeta} - \frac{[(f_{i+1}+f_i)(g_{i+1}-g_i)]^B}{2(\Delta\xi)^2}\right.\right.$$
$$\left.\left. - \frac{[(f_{j+1}+f_j)(g_{j+1}-g_j)]^D}{2(\Delta\eta)^2} - \frac{[(f_{k+1}+f_k)(g_{k+1}-g_k)]^F}{2(\Delta\zeta)^2} - (M+N)\right]\right\}^n \Delta\hat{\mathbf{W}}^n = \Delta\hat{\mathbf{W}}^*$$

where the superscripts B, D, and F denote the terms belonging to the Jacobian coefficient matrices B, D, and F, respectively.

This equation must be solved for the final unknown vector $\Delta\hat{\mathbf{W}}^n$ at (i,j,k). It can be seen that the right hand side of this equation is at the intermediate time level $*$, and that the coefficients on the left hand side are at time level n, and thus known. In addition, the conditions in the planes (N_1, j, k), (i, N_2, k), and (i, j, N_3) at time level $n+1$ are known, because they are the upper boundaries of the computational domain, and the boundary conditions are being treated explicitly.

The marching procedure and the addressing scheme used in this subroutine is analogous to those discussed in the description of subroutine SWUP.

Subroutine SWUP		
Called by	**Calls**	**Purpose**
EXECT		Compute coefficients and source terms, and solve the k-ε equations for the upward LU sweep.

Input

*	CONE	Constant C_1 in the production term of the ε equation.
*	CTWOR	Constant C_{2_r} in formula for C_2.
	DTAU	Time step $\Delta\tau$.
	DUMMY	Distance to the nearest solid wall.
	DW	Dependent variable subvector $\Delta\hat{W}^{n-1}$ from previous time step.
	DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
	E, EL	Turbulent dissipation rate ε at time levels n and $n-1$.
	ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
	JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
	KE, KEL	Turbulent kinetic energy k at time levels n and $n-1$.
	MU	Laminar viscosity μ_l at time level n.
	MUT, MUTL	Turbulent viscosity μ_t at time levels n and $n-1$.
	NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, N_1+1, N_2+1, and N_3+1 for spatially periodic boundary conditions in ξ, η, and ζ.
*	RER	Reference Reynolds number Re_r.
	RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.
	RHOL	Static density ρ at time level $n-1$.
*	SIGE, SIGK	Constants σ_ε and σ_k used in the diffusion term of the ε equation.
*	TFACT	Factor used in computing the k-ε time step.
*	THKE	Parameters θ_1 and θ_2 determining type of time differencing for the k-ε equations.
	VORT	Production rate of turbulent kinetic energy.
	XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
	YPLUSD	Nondimensional distance y^+ from the nearest solid wall.
	ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

DW	Dependent variable subvector $\Delta\hat{W}^*$ from upward LU sweep.

Description

Subroutine SWUP performs the upward LU sweep to solve for the intermediate values of the unknown vector $\Delta\hat{W}^*$ in the k-ε equations. The equation used for the upward sweep is

$$\left\{ \mathbf{I} + \frac{\theta_1 \Delta\tau}{1+\theta_2} \left[\delta_\xi^- A^+ + \delta_\eta^- C^+ + \delta_\zeta^- E^+ - (\delta_\xi B)^- - (\delta_\eta D)^- - (\delta_\zeta F)^- \right] \right\}^n \Delta\hat{\mathbf{W}}^* = \mathrm{RHS}(9.34)$$

where RHS(9.34) represents the right hand side of equation (9.34) in Volume 1. The Jacobian coefficient matrices B, D, and F are made up of terms of the form $f\partial g/\partial\xi$, $f\partial g/\partial\eta$, and $f\partial g/\partial\zeta$, respectively. The terms $(\delta_\xi B)^-$, $(\delta_\eta D)^-$, and $(\delta_\zeta F)^-$ are the backward difference parts of the central differences $\delta_\xi B$, $\delta_\eta D$, and $\delta_\zeta F$, respectively. Thus

$$\delta_\xi B = \delta_\xi \left(f \frac{\partial g}{\partial \xi} \right) = \frac{(f_{i+1} + f_i)(g_{i+1} - g_i) - (f_i + f_{i-1})(g_i - g_{i-1})}{2(\Delta\xi)^2}$$

$$(\delta_\xi B)^- = -\frac{(f_i + f_{i-1})(g_i - g_{i-1})}{2(\Delta\xi)^2}$$

Analogous expressions can be derived for D and F. Expanding the difference terms in the upward sweep equation, we thus get

$$\left\{ \mathbf{I} + \frac{\theta_1 \Delta\tau}{1+\theta_2} \left[\frac{A_i^+ - A_{i-1}^+}{\Delta\xi} + \frac{C_j^+ - C_{j-1}^+}{\Delta\eta} + \frac{E_k^+ - E_{k-1}^+}{\Delta\zeta} + \frac{[(f_i + f_{i-1})(g_i - g_{i-1})]^B}{2(\Delta\xi)^2} \right.\right.$$
$$\left.\left. + \frac{[(f_j + f_{j-1})(g_j - g_{j-1})]^D}{2(\Delta\eta)^2} + \frac{[(f_k + f_{k-1})(g_k - g_{k-1})]^F}{2(\Delta\zeta)^2} \right] \right\}^n \Delta\hat{\mathbf{W}}^* = \mathrm{RHS}(9.34)$$

where the superscripts B, D, and F denote the terms belonging to the Jacobian coefficient matrices B, D, and F, respectively.

This equation must be solved for the intermediate unknown vector $\Delta\hat{\mathbf{W}}^*$ at (i, j, k). It can be seen that the right hand side of this equation, and the coefficients on the left hand side, are at time level n, and thus known. In addition, the conditions in the planes $(1, j, k)$, $(i, 1, k)$, and $(i, j, 1)$ at time level $*$ are known, because they are the lower boundaries of the computational domain, and the boundary conditions are being treated explicitly. Therefore, it is possible to solve this equation by marching point by point from point (2, 2, 2) to point $(N_1 - 1, N_2 - 1, N_3 - 1)$. The conditions in the planes (N_1, j, k), (i, N_2, k), and (i, j, N_3) are known because they are the upper boundaries of the computational domain.

The marching order is unimportant, as long as the march is from the point (2, 2, 2) to point $(N_1 - 1, N_2 - 1, N_3 - 1)$. For example, this could be accomplished using the following pseudo-code:

```
      do 10 i1 = 2,n1-1
      do 10 i2 = 2,n2-1
      do 10 i3 = 3,n3-1
      dw(i1,i2,i3) = function of q(i1,i2,i3), dw(i1-1,i2,i3),
                     dw(i1,i2-1,i3), and dw(i1,i2,i3-1)
10    continue
```

where Q represents the flow field properties and DW is the unknown vector. The Fortran indices I1, I2, and I3 correspond to the grid indices i, j, and k, respectively.

The coding above is correct, but it does not take full advantage of the Cray's vectorization capability. Because it contains two nested do loops, only the innermost loop is vectorized. However, if the marching is done in the direction normal to the diagonal planes of constant $i + j + k$, then the code can be constructed with only one nested do loop, taking better advantage of the Cray's vectorization capability. I.e.,

```
      do 10 iplane = 1,nplane
      do 10 ipoint = 1,npoint
```

```
      dw(ipoint,iplane) = function of q(ipoint,iplane) and dw(ipoint,iplane)
10    continue
```

where NPLANE is the number of diagonal planes in the 3-D computational domain, and NPOINT is the number of interior grid points contained within a diagonal plane. Note that NPOINT varies from plane to plane. It turns out that the points (I1 – 1, I2, I3), (I1, I2 – 1, I3), and (I1, I2, I3 – 1) are all located in the plane IPLANE – 1, and they are known. As the result, the inner loop in the above code can be vectorized over every point in a diagonal plane.

An addressing scheme is needed to translate the indices (IPOINT, IPLANE) to (I1, I2, I3) so that flow properties at (I1,I2,I3) can be recalled in the marching process. There are many ways that this can be accomplished, and in subroutine SWUP a scheme has been devised to compute the I1, I2, and I3 indices from the do loop indices IPOINT and IPLANE. This scheme does not require any special machine-specific routines, and will work for any FORTRAN 77 compiler. Basically, this scheme works as follows:

1. The I1 index of every point in a diagonal plane is stored in the array ILOC(IPOINT).
2. The diagonal line index of every point in a diagonal plane is computed and stored in the array LINE(IPOINT).
3. Inside the nested inner loop, the I1, I2, and I3 indices are then computed from the ILOC(IPOINT) and the LINE(IPOINT) arrays as follows:

```
i1 = iloc(ipoint)
i2 = -i1 + line(ipoint) + 3
i3 = iplane - i1 - i2 + 5
```

Subroutine TBC		
Called by	Calls	Purpose
MAIN		Set time-dependent boundary condition values.

Input

* GTBC1, GTBC2, GTBC3	Time-dependent surface mean flow boundary condition values for the ξ, η, and ζ directions.
IT	Current time step number n.
ITBEG	The time level n at the beginning of a run.
ITEND	Final time step number.
* JBC1, JBC2, JBC3	Surface mean flow boundary condition types for the ξ, η, and ζ directions.
* JTBC1, JTBC2, JTBC3	Flags for type of time dependency for mean flow boundary conditions in the ξ, η, and ζ directions.
NBC	Dimensioning parameter specifying number of boundary conditions per equation.
NEQ	Number of coupled equations being solved, N_{eq}.
* NOUT	Unit number for standard output.
* NTBC	Number of values in tables for general unsteady boundary conditions.
* NTBCA	Time levels at which general unsteady boundary conditions are specified.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.

Output

FBC1, FBC2, FBC3	Point-by-point mean flow boundary condition values for the ξ, η, and ζ directions.
GBC1, GBC2, GBC3	Surface mean flow boundary condition values for the ξ, η, and ζ directions.

Description

Subroutine TBC sets time-dependent mean flow boundary condition values. Two types of time dependency are allowed - general and periodic.

General Time-Dependent Boundary Conditions

General time-dependent boundary conditions are set using linear interpolation on an input table of boundary condition values vs. time level. Thus, the boundary condition value is

$$g^{n+1} = g_t^i + \frac{n + 1 - n_t^i}{n_t^{i+1} - n_t^i}\,(g_t^{i+1} - g_t^i)$$

Here n is the current known time level in the time marching scheme, g_t and n_t represent the input table of boundary condition values vs. time level, and i is the index in the table for which

$$n_t^i \leq n+1 < n_t^{i+1}$$

If $n+1 < n_t^1$, then g^{n+1} is set equal to the first value in the table, g_t^1. Similarly, if $n+1 > n_t^N$, where N is the index of the last entry in the table, then g^{n+1} is set equal to the last value in the table, g_t^N.

In Fortran, $g =$ GBC1, GBC2, or GBC3, $g_t =$ GTBC1, GTBC2, or GTBC3, $n_t =$ NTBCA, and $N =$ NTBC.

Time-Periodic Boundary Conditions

Time-periodic boundary conditions (not to be confused with spatially periodic boundary conditions) are of the form

$$g^{n+1} = g_t^1 + g_t^2 \sin[g_t^3(n+1) + g_t^4]$$

where g_t^1 through g_t^4 are given by the first four elements of GTBC1, GTBC2, or GTBC3.

Remarks

1. An error message is generated and execution is stopped if an invalid type of unsteadiness is requested for the boundary values.

Subroutine TIMSTP		
Called by	Calls	Purpose
MAIN	ISAMAX ISAMIN	Set computational time step.

Input

* CFL	CFL number in IDTAU = 1, 2, 5, 6, 8, and 9 options.
* CFLMIN, CFLMAX	Minimum and maximum CFL numbers allowed in IDTAU = 2 and 6 options.
CHGMAX	Maximum change in absolute value of the dependent variables over previous time step (or NITAVG − 1 time steps if ICTEST = 2), $\Delta \mathbf{Q}_{max}$.
* CHG1, CHG2	Minimum and maximum change, in absolute value, that is allowed in any dependent variable before increasing or decreasing $\Delta\tau$ in IDTAU = 2, 4, and 6 options.
CP, CV	Specific heats c_p and c_v at time level n.
* DT	Time step $\Delta\tau$ in IDTAU = 3 and 4 options.
DTAU	Old computational time step $\Delta\tau$.
* DTF1, DTF2	Factors multiplying or dividing $\Delta\tau$ if solution changes too slowly or quickly in IDTAU = 2, 4, and 6 options.
* DTMIN, DTMAX	Minimum and maximum $\Delta\tau$ allowed in IDTAU = 4 option, or used in IDTAU = 7 option.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
EP2	Maximum allowable numerical value.
ETAX, ETAY, ETAZ, ETAT	Metric coefficients η_x, η_y, η_z, and η_t.
* IDTAU	Flag for time step selection method.
IT	Current time step number n.
ITSEQ	Current time step sequence number.
MU	Effective coefficient of viscosity μ at time level n.
* NDTCYC	Number of time steps per cycle for IDTAU = 7 option.
NEQ	Number of coupled equations being solved, N_{eq}.
* NOUT	Unit number for standard output.
NTOTP	Dimensioning parameter specifying the storage required for a full three-dimensional array (i.e., N1P × N2P × N3P).
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
* RER	Reference Reynolds number Re_r.
RGAS	Gas constant R.
RHO, U, V, W	Static density ρ, and velocities u, v, and w, at time level n.
T	Static temperature T at time level n.
XIX, XIY, XIZ, XIT	Metric coefficients ξ_x, ξ_y, ξ_z, and ξ_t.

ZETAX, ZETAY, ZETAZ, ZETAT — Metric coefficients ζ_x, ζ_y, ζ_z, and ζ_t.

Output

CFL — New CFL number in IDTAU = 2 and 6 options.

DTAU — New computational time step $\Delta\tau$.

Description

Subroutine TIMSTP computes the time step size $\Delta\tau$. The following sections describe the various methods currently available for setting and/or modifying $\Delta\tau$.

IDTAU = 1

This option sets a global (i.e., constant in space) time step $\Delta\tau$ equal to the minimum of the values at each grid point computed from the input parameter CFL(ITSEQ). I.e.,

$$\Delta\tau = (\mathrm{CFL}) \min_{i,j,k}(\Delta\tau_{cfl})$$

where $\Delta\tau_{cfl}$ is the inviscid CFL limit, given in generalized coordinates as (Shang, 1984).

$$\Delta\tau_{cfl} = \left\{ \left| \frac{U}{\Delta\xi} \right| + \left| \frac{V}{\Delta\eta} \right| + \left| \frac{W}{\Delta\zeta} \right| + a\left[\left(\frac{\xi_x}{\Delta\xi} + \frac{\eta_x}{\Delta\eta} + \frac{\zeta_x}{\Delta\zeta} \right)^2 + \left(\frac{\xi_y}{\Delta\xi} + \frac{\eta_y}{\Delta\eta} + \frac{\zeta_y}{\Delta\zeta} \right)^2 + \left(\frac{\xi_z}{\Delta\xi} + \frac{\eta_z}{\Delta\eta} + \frac{\zeta_z}{\Delta\zeta} \right)^2 \right]^{1/2} \right\}^{-1}$$

Here $U = \xi_t + \xi_x u + \xi_y v + \xi_z w$, $V = \eta_t + \eta_x u + \eta_y v + \eta_z w$, and $W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w$ are the contravariant velocities without metric normalization, and $a = \sqrt{\gamma RT}$ is the speed of sound.

IDTAU = 2

For the first time step, this option is identical to the IDTAU = 1 option. After the first time step, however, CFL is modified to keep $\Delta \mathbf{Q}_{max}$, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease CFL may be summarized as follows:

$$\begin{aligned}
\Delta\mathbf{Q}_{max} < \mathrm{CHG1} &\quad\Rightarrow\quad \mathrm{CFL} = \min[(\mathrm{DTF1})(\mathrm{CFL}),\ \mathrm{CFLMAX}] \\
\Delta\mathbf{Q}_{max} > \mathrm{CHG2} &\quad\Rightarrow\quad \mathrm{CFL} = \max[\mathrm{CFL}/\mathrm{DTF2},\ \mathrm{CFLMIN}] \\
\Delta\mathbf{Q}_{max} > 0.15 &\quad\Rightarrow\quad \mathrm{CFL} = \mathrm{CFL}/2
\end{aligned}$$

The time step $\Delta\tau$ is then set using the same formulas as in the IDTAU = 1 option.

IDTAU = 3

This option sets a global (i.e., constant in space) time step $\Delta\tau$ equal to the input parameter DT(ITSEQ).

IDTAU = 4

For the first time step, this option is identical to the IDTAU = 3 option. After the first time step, however, $\Delta\tau$ is modified to keep $\Delta\mathbf{Q}_{max}$, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease $\Delta\tau$ may be summarized as follows:

$$\Delta Q_{max} < \text{CHG1} \Rightarrow \Delta\tau = \min[(\text{DTF1})\Delta\tau,\ \text{DTMAX}]$$
$$\Delta Q_{max} > \text{CHG2} \Rightarrow \Delta\tau = \max[\Delta\tau/(\text{DTF2}),\ \text{DTMIN}]$$
$$\Delta Q_{max} > 0.15 \Rightarrow \Delta\tau = \Delta\tau/2$$

IDTAU = 5

This option sets a local (i.e., varying in space) time step $\Delta\tau$ computed at each grid point from the input parameter CFL(ITSEQ). I.e., at each grid point,

$$\Delta\tau = (\text{CFL})\Delta\tau_{cfl}$$

where $\Delta\tau_{cfl}$ is given above in the description of the IDTAU = 1 option.

IDTAU = 6

For the first time step, this option is identical to the IDTAU = 5 option. After the first time step, however, CFL is modified to keep ΔQ_{max}, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease CFL are the same as in the IDTAU = 2 option.

IDTAU = 7

This option sets a global (i.e., constant in space) time step $\Delta\tau$ with logarithmic cycling. The formula used is

$$\Delta\tau = \Delta\tau_{min}\left(\frac{\Delta\tau_{max}}{\Delta\tau_{min}}\right)^{N/(N_{cyc}-1)}$$

where $N = \text{mod}(n-1, N_{cyc})$ and n is the current known time level. The time step $\Delta\tau$ is thus cycled repeatedly between $\Delta\tau_{min}$ and $\Delta\tau_{max}$ every N_{cyc} time steps. The values of $\Delta\tau_{min}$, $\Delta\tau_{max}$, and N_{cyc} are given by the input parameters DTMIN, DTMAX, and NDTCYC.

IDTAU = 8

This option sets a local (i.e., varying in space) time step $\Delta\tau$ computed at each grid point using the procedure of Knight and Choi (1989). The inviscid CFL limit $\Delta\tau_{cfl}$ is first computed separately for each computational coordinate direction. Thus, at each grid point,

$$(\Delta\tau_{cfl})_\xi = \left[\left|\frac{U}{\Delta\xi}\right| + a\frac{\sqrt{\xi_x^2+\xi_y^2+\xi_z^2}}{\Delta\xi}\right]^{-1}$$

$$(\Delta\tau_{cfl})_\eta = \left[\left|\frac{V}{\Delta\eta}\right| + a\frac{\sqrt{\eta_x^2+\eta_y^2+\eta_z^2}}{\Delta\eta}\right]^{-1}$$

$$(\Delta\tau_{cfl})_\zeta = \left[\left|\frac{W}{\Delta\zeta}\right| + a\frac{\sqrt{\zeta_x^2+\zeta_y^2+\zeta_z^2}}{\Delta\zeta}\right]^{-1}$$

Here $U = \xi_t + \xi_x u + \xi_y v + \xi_z w$, $V = \eta_t + \eta_x u + \eta_y v + \eta_z w$, and $W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w$ are the contravariant velocities without metric normalization, and $a = \sqrt{\gamma RT}$ is the speed of sound.

A preliminary value of $\Delta\tau$ is then defined at each grid point using the input parameter CFL(ITSEQ).

$$\Delta\tau_0 = (\mathrm{CFL}) \min[(\Delta\tau_{cfl})_\xi, (\Delta\tau_{cfl})_\eta, (\Delta\tau_{cfl})_\zeta]$$

The final value of $\Delta\tau$ is then defined at each grid point as

$$\Delta\tau = \max[\Delta\tau_0, (\Delta\tau_{cfl})_\xi]$$

Knight and Choi found that using this definition for $\Delta\tau$, rather than simply setting $\Delta\tau = \Delta\tau_0$, resulted in faster convergence for problems with refined grid regions. This formulation assumes that flow is generally in the ξ direction.

IDTAU = 9

This option is similar to the IDTAU = 8 option. The only difference is a viscous correction added to the definitions of the inviscid CFL limits, similar to that used by Cooper (1987). The inviscid CFL limits are now defined at each grid point as:

$$(\Delta\tau_{cfl})_\xi = \left[\left| \frac{U}{\Delta\xi} \right| + a\frac{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}{\Delta\xi} + \frac{2}{Re_r}\frac{\mu}{\rho}\frac{\xi_x^2 + \xi_y^2 + \xi_z^2}{(\Delta\xi)^2} \right]^{-1}$$

$$(\Delta\tau_{cfl})_\eta = \left[\left| \frac{V}{\Delta\eta} \right| + a\frac{\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}}{\Delta\eta} + \frac{2}{Re_r}\frac{\mu}{\rho}\frac{\eta_x^2 + \eta_y^2 + \eta_z^2}{(\Delta\eta)^2} \right]^{-1}$$

$$(\Delta\tau_{cfl})_\zeta = \left[\left| \frac{W}{\Delta\zeta} \right| + a\frac{\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}}{\Delta\zeta} + \frac{2}{Re_r}\frac{\mu}{\rho}\frac{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}{(\Delta\zeta)^2} \right]^{-1}$$

The rest of the procedure for computing $\Delta\tau$ is the same as in the IDTAU = 8 option.

Remarks

1. In ΔQ_{max}, used in the IDTAU = 2, 4, and 6 options, the change in E_T has been divided by $R/(\gamma_r - 1) + 1/2$. This is equivalent to dividing the dimensional value $\overline{E}_T$ by

$$E_{T_r} = \frac{\rho_r \overline{R} T_r}{\gamma_r - 1} + \frac{\rho_r u_r^2}{2}$$

   This makes the change in total energy the same order of magnitude as the other conservation variables.
2. An error message is generated and execution is stopped if an illegal time step selection option is requested.
3. A warning message is printed with the IDTAU = 2, 4, and 6 options if $\Delta\tau$ or the CFL number is cut in half because $\Delta Q_{max} > 0.15$.
4. The Cray search routine ISAMAX is used in computing the maximum value of ΔQ_{max} for all the equations.

Subroutine TREMAIN (CPUREM)		
Called by	Calls	Purpose
MAIN		Get CPU time remaining for the job.

Input

None.

Output

CPUREM — Amount of CPU time remaining, in seconds.

Description

Subroutine TREMAIN computes the amount of CPU time remaining for the current job, in seconds.

Remarks

1. TREMAIN is a Cray Fortran library routine (Cray Research, Inc., 1989a).

Subroutine TURBBL		
Called by	**Calls**	**Purpose**
INITC KEINIT MAIN	BLIN BLOUT	Manage computation of turbulence parameters using Baldwin-Lomax algebraic model.

Input

	CP	Specific heat c_p.
	EP1	Minimum allowable numerical value.
*	KBC1, KBC2, KBC3	Boundary types for the ξ, η, and ζ directions.
	LWSET	Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
	MU, LA, KT	Laminar coefficient of viscosity μ_l, laminar second coefficient of viscosity λ_l, and laminar coefficient of thermal conductivity k_l.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	PRR	Reference Prandtl number Pr_r.
*	PRT	Turbulent Prandtl number Pr_t, or, if PRT ≤ 0, a flag indicating the use of a variable turbulent Prandtl number.
	U, V, W	Velocities u, v, and w.

Output

LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries, if not set in input.
MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k.

Description

Subroutine TURBBL manages the computation of the effective coefficient of viscosity, second coefficient of viscosity, and coefficient of thermal conductivity using the algebraic eddy viscosity model of Baldwin and Lomax (1978). It is called from MAIN at the end of each step from time level n to $n+1$, after the governing flow equations have been solved. The Fortran variables RHO, U, etc., are thus at the $n+1$ level. The effective viscosity coefficient to be computed will therefore also be at the $n+1$ level. This, of course, becomes the known n level for the next time step.

The steps involved in computing the effective coefficients are as follows:

1. Initialize the array for storing the turbulent viscosity μ_t to zero.
2. Determine wall locations by checking for zero velocity at the boundaries, unless wall locations are user-specified via the input LWALL or IWALL parameters, or unless boundary types are specified using the KBC parameters.
3. Call BLOUT to compute $(\mu_t)_{outer}$ at each interior grid point.
4. Call BLIN to compute $(\mu_t)_{inner}$ at each interior grid point. BLIN then sets $\mu_t = \min[(\mu_t)_{inner}, (\mu_t)_{outer}]$.

5. Define the necessary effective coefficients as follows:

$$\mu = \mu_l + \mu_t$$
$$\lambda = \lambda_l + \lambda_t$$
$$k = k_l + k_t$$

where $\lambda_t = -2\mu_t/3$, and k_t is computed using Reynolds analogy as

$$k_t = \frac{\mu_t c_p}{Pr_t}$$

The turbulent Prandtl number is either a constant specified in the input, or a variable computed using equation (9.17) of Volume 1.

Remarks

1. In the Fortran equation for the effective thermal conductivity, the factor PRR = Pr_r is necessary for proper nondimensionalization of k_t.

Subroutine TURBCH		
Called by	Calls	Purpose
MAIN	EXECT PRODCT YPLUSN	Manage computation of turbulence parameters using the Chien k-ε model.

Input

	CP	Specific heat c_p.
	EP1	Minimum allowable numerical value.
*	KBC1, KBC2, KBC3	Boundary types for the ξ, η, and ζ directions.
	LWSET	Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
	MU, LA, KT	Laminar coefficient of viscosity μ_l, laminar second coefficient of viscosity λ_l, and laminar coefficient of thermal conductivity k_l.
*	NTKE	Number of k-ε iterations per mean flow iteration.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
	PRR	Reference Prandtl number Pr_r.
*	PRT	Turbulent Prandtl number Pr_t, or, if PRT ≤ 0, a flag indicating the use of a variable turbulent Prandtl number.
	U, V, W	Velocities u, v, and w at time level n.

Output

LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries, if not set in input.
MU, LA, KT	Effective coefficient of viscosity μ, effective second coefficient of viscosity λ, and effective coefficient of thermal conductivity k.

Description

Subroutine TURBCH manages the computation of the effective coefficient of viscosity, second coefficient of viscosity, and coefficient of thermal conductivity using the low Reynolds number k-ε two-equation turbulence model of Chien (1982). The k-ε equations are uncoupled from the mean flow equations, lagged in time and solved separately. This allows maximum modularity in turbulence modeling.

For each step from time level n to $n+1$, the mean flow equations are solved first, using a time step $\Delta\tau$. The k-ε equations are then solved, using NTKE time steps with a time step size of TFACT($\Delta\tau$).

The steps involved in computing the effective coefficients are as follows:

1. Determine wall locations by checking for zero velocity at the boundaries, unless wall locations are user-specified via the input LWALL or IWALL parameters, or unless boundary types are specified using the KBC parameters.
2. Call YPLUSN to compute the distance to the nearest solid wall and y^+. To save storage, the distance is returned in the Fortran variable DUMMY.

3. Call PRODCT to compute the production rate of turbulent kinetic energy. To save storage space, the production rate is returned in the Fortran variable VORT.
4. Call EXECT to advance the k-ε equations in time using a time step of TFACT($\Delta\tau$).
5. Repeat steps 2-4 NTKE times.
6. Define the necessary effective coefficients as follows:

$$\mu = \mu_l + \mu_t$$
$$\lambda = \lambda_l + \lambda_t$$
$$k = k_l + k_t$$

where $\lambda_t = -2\mu_t/3$, and k_t is computed using Reynold's analogy as

$$k_t = \frac{\mu_t c_p}{Pr_t} Pr_r$$

The turbulent Prandtl number is either a constant specified in the input, or a variable computed using equation (9.17) of Volume 1.

Remarks

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the distance to the nearest wall. The array is filled in subroutine YPLUSN.
2. The Fortran array VORT, from the common block TURB1, is used to store the values of the production rate of turbulent kinetic energy. The array is filled in subroutine PRODCT.
3. For equal mean flow and k-ε time steps, use TFACT = 1/NTKE.

Subroutine UPDATE		
Called by	Calls	Purpose
EXEC		Update flow variables after each ADI sweep.

Input

IBASE, ISTEP	Base index and multiplication factor used in computing one-dimensional index for three-dimensional array.
* IHSTAG	Flag for constant stagnation enthalpy option.
IV	Index in the "vectorized" direction, i_v.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
NPTS	Number of grid points in the sweep direction, N.
NR, NRU, NRV, NRW, NET	Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and E_T.
RHO, U, V, W, ET	Static density ρ, velocities u, v, and w, and total energy E_T at time level n.
S	Computed solution subvector, $\Delta\hat{\mathbf{Q}}$.

Output

RHOL, UL, VL, WL, ETL	Static density ρ, velocities u, v, and w, and total energy E_T at end of current ADI sweep.

Description

Subroutine UPDATE computes the primitive flow variables from the dependent variables $\Delta\hat{\mathbf{Q}}$ after each ADI sweep. For the first sweep the formulas are

$$\rho^* = \rho^n + J\Delta\hat{Q}_1^*$$

$$u^* = \frac{1}{\rho^*}(\rho^n u^n + J\Delta\hat{Q}_2^*)$$

$$v^* = \frac{1}{\rho^*}(\rho^n v^n + J\Delta\hat{Q}_3^*)$$

$$w^* = \frac{1}{\rho^*}(\rho^n w^n + J\Delta\hat{Q}_4^*)$$

$$E_T^* = E_T^n + J\Delta\hat{Q}_5^*$$

where $\Delta\hat{Q}_1$ through $\Delta\hat{Q}_5$ are the dependent variables in delta form for the five governing equations.[33] For the second ADI sweep, the superscript * should be changed to ** on ρ, u, v, w, E_T, and $\Delta\hat{Q}$. For the third ADI sweep, the superscript * should be changed to $n+1$ on ρ, u, v, w, and E_T, and to n on $\Delta\hat{Q}$.

Remarks

1. This subroutine uses one-dimensional addressing of three-dimensional arrays, as described in Section 2.3.

[33] These formulas are written for non-constant stagnation enthalpy. If constant stagnation enthalpy is assumed, there will be only four equations.

Subroutine UPDTKE		
Called by	Calls	Purpose
EXECT		Update k and ε after each ADI sweep.

Input

DW	Dependent variable subvector $\Delta\hat{\mathbf{W}}^n$ from downward LU sweep.
DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
E	Turbulent dissipation rate ε at time level n.
* FBCT1, FBCT2, FBCT3	Point-by-point k-ε boundary condition values for the ξ, η, and ζ directions.
* IBCT1, IBCT2, IBCT3	Point-by-point k-ε boundary condition types for the ξ, η, and ζ directions.
JI	Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
KBCPER	Flags for spatially periodic boundary conditions in the ξ, η, and ζ directions; 0 for non-periodic, 1 for periodic.
KE	Turbulent kinetic energy k at time level n.
NPT1, NPT2, NPT3	N_1, N_2, and N_3 for non-periodic boundary conditions, $N_1 + 1$, $N_2 + 1$, and $N_3 + 1$ for spatially periodic boundary conditions in ξ, η, and ζ.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
RHO	Static density ρ at time level n.

Output

E	Turbulent dissipation rate ε at time level $n + 1$. and n.
KE	Turbulent kinetic energy k at time level $n + 1$.

Description

Subroutine UPDTKE computes the primitive flow variables k and ε from the dependent variables $\Delta\hat{\mathbf{W}}^n$ after a complete time step. The formulas are

$$k^{n+1} = \frac{1}{\rho^{n+1}}\left(\rho^{n+1} k^n + J\Delta\hat{W}_1^n\right)$$

$$\varepsilon^{n+1} = \frac{1}{\rho^{n+1}}\left(\rho^{n+1} \varepsilon^n + J\Delta\hat{W}_2^n\right)$$

where $\Delta\hat{W}_1$ and $\Delta\hat{W}_2$ are the dependent variables in delta form for the k-ε equations.

Subroutine UPDTKE also explicitly computes the k and ε values on the computational boundaries using the specified boundary conditions, as described below.

No Change From Initial Conditions, $\Delta k = 0$ and/or $\Delta \varepsilon = 0$

Values for k and ε are simply not updated. Therefore, their values on the boundaries remain the same as their initial or restart values.

Specified values, $k = f$ and/or $\varepsilon = f$

Values of k and ε are simply set equal to the specified values.

Specified Two-Point Gradient in Coordinate Direction, $\partial k/\partial \phi = f$ and/or $\partial \varepsilon/\partial \phi = f$

Applying $\partial k/\partial \phi = f$ at the $\xi = 0$ boundary, and using two-point one-sided differencing, gives

$$k_{1,j,k} = k_{2,j,k} - f\Delta\xi$$

At the $\xi = 1$ boundary,

$$k_{N_1,j,k} = k_{N_1-1,j,k} + f\Delta\xi$$

Analogous equations can easily be written for the η and ζ boundaries, and for $\partial \varepsilon/\partial \phi = f$.

Specified Three-Point Gradient in Coordinate Direction, $\partial k/\partial \phi = f$ and/or $\partial \varepsilon/\partial \phi = f$

Applying $\partial k/\partial \phi = f$ at the $\xi = 0$ boundary, and using three-point one-sided differencing, gives

$$k_{1,j,k} = \frac{(4k_{2,j,k} - k_{3,j,k} - 2f\Delta\xi)}{3}$$

At the $\xi = 1$ boundary,

$$k_{N_1,j,k} = \frac{(k_{N_1-1,j,k} - k_{N_1-2,j,k} + 2f\Delta\xi)}{3}$$

Analogous equations can easily be written for the η and ζ boundaries, and for $\partial \varepsilon/\partial \phi = f$.

Linear Extrapolation

Linearly extrapolating from the interior points for k at the $\xi = 0$ boundary gives

$$k_{1,j,k} = 2k_{2,j,k} - k_{3,j,k}$$

At the $\xi = 1$ boundary,

$$k_{N_1,j,k} = 2k_{N_1-1,j,k} - k_{N_1-2,j,k}$$

Analogous equations can easily be written for the η and ζ boundaries, and for linear extrapolation of ε.

Remarks

1. The "no change from initial conditions" boundary condition is applied simply by non-execution of the other boundary conditions.
2. Periodic boundary conditions are updated by setting the values of k and ε at the lower boundary equal to the corresponding values at the upper boundary.
3. When a specified gradient or linear extrapolation boundary condition is used, k and/or ε at the boundary is forced to be positive by using the absolute value. This is done to avoid unphysical negative values that could result from poor initial profiles for k and/or ε.

Subroutine VORTEX		
Called by	Calls	Purpose
BLIN BLOUT OUTPUT YPLUSN		Compute magnitude of total vorticity.

Input

DXI, DETA, DZETA	Computational grid spacing $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$.
ETAX, ETAY, ETAZ	Metric coefficients η_x, η_y, and η_z.
* N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
U, V, W	Velocities u, v, and w.
XIX, XIY, XIZ	Metric coefficients ξ_x, ξ_y, and ξ_z.
ZETAX, ZETAY, ZETAZ	Metric coefficients ζ_x, ζ_y, and ζ_z.

Output

VORT	Total vorticity magnitude.

Description

Subroutine VORTEX computes the magnitude of the total vorticity vector. This is defined as

$$|\vec{\Omega}| = \left[\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right)^2 \right]^{1/2}$$

Using the chain rule, these can be rewritten in generalized nonorthogonal coordinates as

$$\begin{aligned} |\vec{\Omega}| = \Big\{ & [(\xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta) - (\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta)]^2 \\ & + [(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta) - (\xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta)]^2 \\ & + [(\xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta) - (\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta)]^2 \Big\}^{1/2} \end{aligned}$$

At interior points, the centered difference formula presented in Section 5.0 of Volume 1 is used to numerically compute the derivatives in the above equations. At boundary points, second-order one-sided difference formulas are used.

Subroutine WHENFLT (N,V,INC,VALUE,INDEX,NVAL)		
Called by	Calls	Purpose
BLOUT		Find all indices in an array whose elements are less than a specified value.

Input

N	Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).
V	Vector to be searched.
INC	Skip distance between elements of V. For contiguous elements, INC = 1.
VALUE	Value to be searched for in the vector V.

Output

INDEX	Vector of indices specifying which elements of V are less than VALUE.
NVAL	Number of values in V that are less than VALUE.

Description

Subroutine WHENFLT finds all indices in an array whose elements are less than a specified value.

Remarks

1. WHENFLT is a Cray search routine (Cray Research, Inc., 1989b).

Subroutine YPLUSN		
Called by	Calls	Purpose
INITC KEINIT TURBCH	VORTEX	Compute the distance to the nearest solid wall.

Input

*	LWALL1, LWALL2, LWALL3	Flags specifying wall locations for ξ, η, and ζ boundaries.
	MU	Effective coefficient of viscosity μ.
*	N1, N2, N3	Number of grid points N_1, N_2, and N_3, in the ξ, η, and ζ directions.
*	RER	Reference Reynolds number Re_r.
	RHO	Static density ρ at time level n.
	X, Y, Z	Cartesian coordinates x, y, and z.

Output

DUMMY	Distance to the nearest solid wall.
YPLUSD	Nondimensional distance y^+ from the nearest solid wall.

Description

Subroutine YPLUSN computes the minimum distance to the nearest solid wall and y^+ for every grid point in the computational domain. The steps involved are as followed:

1. Call VORTEX to compute total vorticity magnitude .
2. For every grid point in the computational domain,
   3. Compute the distance to each solid wall, and the corresponding wall values of the total vorticity magnitude, laminar viscosity, and density.
   4. Identify the nearest solid wall and select the corresponding distance to the wall y_n, the wall total vorticity magnitude $|\Omega_{wall}|$, the wall laminar viscosity μ_{wall}, and the wall density ρ_{wall}.
   5. Compute y^+ using

$$y^+ = y_n \sqrt{\frac{Re_r |\Omega_{wall}| \rho_{wall}}{\mu_{wall}}}$$

Remarks

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the distance to the nearest solid wall.
2. This subroutine will return very large values for YPLUSD and DUMMY if none of the boundaries are solid walls.

REFERENCES

Baldwin, B. S., and Lomax, H. (1978) "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257.

Beam, R. M., and Warming, R. F. (1978) "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, No. 4, pp. 393-402.

Briley, W. R., and McDonald, H. (1977) "Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, pp. 373-397.

Cebeci, T., and Bradshaw, P. (1984) *Physical and Computational Aspects of Convective Heat Transfer,* Springer-Verlag, New York.

Chen, S. C., and Schwab, J. R. (1988) "Three-Dimensional Elliptic Grid Generation Technique with Application to Turbomachinery Cascades," NASA TM 101330.

Chien, K. Y. (1982) "Prediction of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," AIAA Journal, Vol. 20, No. 1, pp. 33-38.

Cooper, G. K. (1987) "The PARC Code: Theory and Usage," AEDC-TR-87-24.

Cray Research, Inc. (1988) *UPDATE Reference Manual,* Publication Number SR-0013.

Cray Research, Inc. (1989a) *Volume 1: UNICOS Fortran Library Reference Manual,* Publication Number SR-2079.

Cray Research, Inc. (1989b) *Volume 3: UNICOS Math and Scientific Library Reference Manual,* Publication Number SR-2081.

Cray Research, Inc. (1990) *CF77 Compiling System, Volume 1: Fortran Reference Manual,* Publication Number SR-3071.

Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W. (1979) *LINPACK User's Guide* SIAM, Philadelphia.

Faux, I. D., and Pratt, M. J. (1979) *Computational Geometry for Design and Manufacture,* Ellis Horwood Limited, John Wiley & Sons, Chichester, England.

Hesse, W. J., and Mumford, N. V. S. (1964) *Jet Propulsion for Aerospace Applications* Pitman Publishing Corporation, New York.

Jameson, A., Schmidt, W., and Turkel, E. (1981) "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259.

Kernighan, B. W., and Plauger, P. J. (1978) *The Elements of Programming Style,* McGraw-Hill Book Company, New York.

Kleinstein, G. (1967) "Generalized Law of the Wall and Eddy-Viscosity Model for Wall Boundary Layers," AIAA Journal, Vol. 5, No. 8, pp. 1402-1407.

Knight, C. J., and Choi, D. (1989) "Development of a Viscous Cascade Code Based on Scalar Implicit Factorization," AIAA Journal, Vol. 27, No. 5, pp. 581-594.

Launder, B. E., and Priddin, C. H. (1973) "A Comparison of Some Proposals for the Mixing Length Near a Wall," International Journal of Heat and Mass Transfer, Vol. 16, pp. 700-702.

Pulliam, T. H. (1986b) "Artificial Dissipation Models for the Euler Equations," AIAA Journal, Vol. 24, No. 12, pp. 1931-1940.

Roberts, G. O. (1971) "Computational Meshes for Boundary Layer Problems," Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics, *Lecture Notes in Physics,* Vol. 8, Springer-Verlag, New York, pp. 171-177.

Shang, J. S. (1984) "Numerical Simulation of Wing-Fuselage Aerodynamic Interaction," AIAA Journal, Vol. 22, No. 10, pp. 1345-1353.

Spalding, D. B. (1961) "A Single Formula for the Law of the Wall," Journal of Applied Mechanics, Vol. 28, pp. 455-457.

Steger, J. L. (1978) "Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries," AIAA Journal, Vol. 16, No. 7, pp. 679-686.

Towne, C. E., Schwab, J. R., Benson, T. J., and Suresh, A. (1990) "PROTEUS Two-Dimensional Navier-Stokes Computer Code - Version 1.0, Volumes 1-3," NASA TM's 102551-3.

White, F. M. (1974) *Viscous Fluid Flow,* McGraw-Hill Book Company, New York.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1993

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Proteus Three-Dimensional Navier-Stokes Computer Code–Version 1.0
Volume 3–Programmer's Reference

5. FUNDING NUMBERS
WU–505–62–52

6. AUTHOR(S)
Charles E. Towne, John R. Schwab, and Trong T. Bui

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E–8110

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM–106341

11. SUPPLEMENTARY NOTES
Responsible person, Charles E. Towne, (216) 433–5851.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 34

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A computer code called *Proteus 3D* has been developed to solve the three-dimensional, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized. The governing equations are solved in generalized nonorthogonal bodyfitted coordinates, by marching in time using a fully-coupled ADI solution procedure. The boundary conditions are treated implicitly. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model. The thin-layer or Euler equations may also be solved. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used. Several time step options are available for convergence acceleration. The documentation is divided into three volumes. This is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

14. SUBJECT TERMS
Navier-Stokes; Computational fluid dynamics; Viscous flow; Compressible flow

15. NUMBER OF PAGES
284

16. PRICE CODE
A13

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	